Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Інститут/факультет «Інститут прикладного системного аналізу»
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною (освітньо-науковою) програмою

Спеціальність (спеціалізація) 122 – комп'ютерні науки та інформаційні технології (Інформаційні системи і технології проектування)
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І. Петренко
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ на магістерську дисертацію студенту
Аззузу Іскандару Джабра
(прізвище, ім'я, по батькові)

1. Тема дисертації Узгодження протоколів керування мережею "розумних" речей для хмарного Thing Speak та CAN internet of things., науковий керівник дисертації Кірюша Богдан Анатолійович, к.т.н., доц. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «27» березня 2018 р. № 1028-с

2. Термін подання студентом дисертації 10 травня 2018р.

3. Об'єкт дослідження протоколи, що використовуються для побудови мереж розумних речей IoT систем

4. Предмет дослідження засоби та методи узгодження керування мережею IoT пристроїв

5. Перелік завдань, які потрібно розробити

    1. Провести дослідження концепції побудови IoT інфраструктур.
    2. Провести аналіз протоколів передачі даних, що застосовуються в мережах IoT пристроїв
    3. Провести огляд хмарних платформ

4. Провести аналіз засобів керування IoT пристроями та мережею CAN

5. Спроектувати систему для узгодження протоколів керування пристроїв в мережі CAN та хмарного ThingSpeak.

6. Орієнтовний перелік ілюстративного матеріалу презентація на тему:_____

«Узгодження протоколів керування мережею "розумних" речей для хмарного Thing Speak та CAN internet of things»_____

7. Орієнтовний перелік публікацій

Аззуз І.А., Бужак Ю.Ю., Шеренковський А.О. Платформи для Інтернету речей // XIV міжнародна науково-практична конференція. – 2016. – №12. – С. 241-246.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розробка стартап-проекту			

9. Дата видачі завдання 01.02.2018

Календарний план

10. Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.02.2018	
2	Збір інформації та аналіз літератури	15.02.2018	
3	Проведення огляду архітектури IOT	28.02.2018	
4	Аналіз протоколів в мережах ОТ	11.03.2018	
5	Формалізація задачі узгодження керування «розумними» пристроями	25.03.2018	
6	Реалізація поставленої задачі	13.04.2018	
7	Аналіз результатів моделювання	25.04.2018	
8	Оформлення дипломної роботи	30.04.2018	
9	Отримання допуску до захисту та подача роботи в ДЕК	09.05.2018	

Студент _____ І.Д. Аззуз
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Б.А. Кірюша
(підпис) (ініціали, прізвище)

РЕФЕРАТ НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ

виконану на тему: Узгодження протоколів керування мережею "розумних" речей для хмарного Thing Speak та CAN internet of things.

студентом: Аззузом Іскандаром Джабра

Робота виконана на 84 сторінках, містить 21 ілюстрацій, 27 таблиці. При підготовці використовувалась література з 21 джерела.

Актуальність теми

З розвитком IoT індустрії з’явилось безліч технологій, протоколів та засобів для керування мережею «розумних» речей. У зв’язку з цим все більш необхідні механізми узгодження різних стандартів, пристроїв, способу їх зв’язку.

Таким чином дослідження даних технологій, протоколів та засобів узгодження керування мережами «розумних речей» є актуальним напрямком дослідження, саме на сьогоднішній день, коли індустрія швидко змінюється, а єдиного підходу до управління IoT пристроїв немає.

Мета та задачі дослідження

Метою дослідження є проведення аналізу, систематизації, порівняння протоколів передачі даних в IoT системах, огляд засобів керування «розумними речами» в мережах CAN та в цілому.

Результатом проведених досліджень є практична частина роботи, що становить собою побудову моделі узгодження протоколів мережі CAN та хмарного ThingSpeak.

Рішення поставлених завдань та досягнуті результати

У даній роботі було розглянуто основні структурні шари, на які поділяють архітектуру IoT систем. У роботі проведено аналіз протоколів, що застосовуються для побудови IoT мереж, таких як WiFi, Zigbee, 6LOWPAN, CAN, на прикладному рівні розглянуто MQTT, COAP, HTTP REST. Також проведено огляд хмарних платформ Google Cloud Platform IoT, Azure IoT, AWS, ThingSpeak. Надалі розглянуто засоби керування IoT пристроями в цілому, та «розумними» речами в мережах CAN.

В рамках практичної частини було побудовано модель узгодження протоколів керування в мережі CAN IoT та платформи ThingSpeak. В якості IoT шлюзу було обрано Raspberry PI 3, який був об’єднаний в CAN мережу з датчиком, побудованим на апаратній платформі Arduino Uno з сенсором температури та вологості. Реалізовано програму для узгодження протоколу CANOpen, який використовується між датчиком та шлюзом, та HTTP Rest, що застосовується для передачі даних на хмарний ThingSpeak через WiFi або Ethernet.

Результати роботи демонструють, що дане рішення може використовуватись в мережах IoT, особливо, коли є вимоги до надійного зв’язку між пристроями.

Об’єкт досліджень

Протоколи, що використовуються для побудови мереж розумних речей IoT систем.

Предмет досліджень

Засоби та методи узгодження керування мережею IoT пристроїв.

Методи досліджень

Для вирішення проблеми в даній роботі використовуються методи аналізу і синтезу, системного аналізу, порівняння, логічного узагальнення результатів.

Наукова новизна

Наукова новизна роботи полягає у створенні нового підходу в керуванні мережею «розумних» речей.

Ключові слова

Протоколи керування, архітектура IoT, «розумні» речі, CAN, ThingSpeak, Raspberry, Arduino

ABSTRACT

on master's thesis

on topic: "The CAN internet of things and Thing speak protocols synchronization"

Work carried out on 84 pages, containing 21 figures, 27 tables. The paper was written with references to 21 different sources.

Topicality

With the growing popularity of IoT in the area of information technology, many manufacturers began to develop and present their solutions to devices, protocols and interoperability technologies that could become the basis for IoT.

At the moment, there are many IoT devices, data transfer protocols, technologies and tools on the market for building such systems. The main advantages of the system of "smart" devices is the ability to automate their work and remote control.

Thus, studying the data of technologies, protocols and tools for managing the networks of "smart things" is an actual direction of research, as it is today, at a time when everything is fast changing, and there are no common approaches to manage IoT devices.

Purpose

The purpose of this work is organizing, analysis, compare of data transfer protocols that are used in IoT systems. As well as an overview of smart things in CAN and in general.

The result of the research is a practical part of the work, which demonstrates a model, that coordinate management protocols for CAN network devices and cloud-based ThingSpeak.

The solution of the tasks and results

This dissertation explores the main structural layers of IoT systems. In this work the protocols for IoT networks was analyzed, such as WiFi, Zigbee, 6LOWPAN, CAN, also MQTT, COAP, HTTP REST on application level. The overview of the Google Cloud Platform IoT, Azure IoT, AWS, ThingSpeak cloud-

based platforms is done. Also, the tools for management IoT devices and CAN-based networks are analyzed.

The main focus of the experiments was to build the model, that coordinate management protocols for CAN network devices and cloud-based ThingSpeak. Raspberry PI 3 was chosen to emulate IoT gateway. It was connected in CAN network with sensor of temperature and humidity based on Arduino Uno platform. The program for coordination CanOpen protocol between sensor and gateway and HTPP Rest for data transferring between gateway and cloud ThingSpeak was implemented.

This result highlights that current approach can be used in IoT networks, especially, when there are requirements for high reliability and quality of data transmission.

The object of research

Protocols that are used for development of “smart” thing IoT networks.

The subject of research

Tools and methods to coordinate management of IoT devices networks.

Scientific novelty

Scientific novelty lies in the innovative approach to manage that coordinate management protocols for CAN network devices and cloud-based ThingSpeak.

Key words

Management protocols, IoT architecture, “smart” things, CAN, ThingSpeak, CanOpen, Raspberry, Arduino.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IoT	Internet of Things, Інтернет речей
Framework	готовий до використання комплекс програмних рішень
ПЗ	програмне забезпечення
CAN	Controller Area Network
BLE	Bluetooth Low Energy
ОС	операційна система
HTTP	Hyper Text Transfer Protocol
MQTT	Message Queue Telemetry Transport
REST	Representational State Transfer
COAP	Constrained Application Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol

ВСТУП

За останні роки спостерігається значний ріст та розвиток інформаційних та комунікаційних технологій. Все що оточує людину в її побутовому житті стає «розумним». В зв’язку з цим індустрія IoT стрімко набирає популярності як в галузях автоматизації промисловісті, медицини, торгівлі, екології, траспорту так і побутового життя людини, де системи «розумних» будинків знайшли своє призначення. Інтернет речей доволі нова парадигма, що намагається визначити найближче майбутнє, де всі об’єкти повсякденного життя будуть обладнані мікроконтролерами, передавачами для комунікації один з одним та з користувачем. Таким чином IoT описується як розподілена мережа фізичних об’єктів, що можуть спілкуватись з іншими пристроями. Інтернет речей забезпечує легкий доступ та взаємодію з різноманітними пристроями: побутова техніка, камери спостереження, датчики, виконавчі пристрої, дисплеї, транспортні засоби тощо.

IoT впливає на кожен бізнес. Інтернет речей змінить типи пристроїв, які підключаються до систем компанії. Він допоможе підприємству підняти корисну ефективність, поліпшити операції та підвищити рівень задоволеності клієнтів. IoT також матиме великий вплив на життя людей, покращить громадську безпеку, транспорт та охорону здоров'я. Хоча є багато способів, якими Інтернет речейй міг би вплинути на соціум та бізнес, є принаймні три основні переваги IoT, що вплинуть на будь-який бізнес:

- Комунікацію
- Контроль
- Економію коштів

Однак, такі гетерогенні області застосування ускладнюють побудову єдиної архітектури IoT, через її складність та новизну.

З’явилось безліч технологій, протоколів та засобів для керування мережею «розумних» речей. У зв’язку з цим все більш необхідні механізми узгодження різних стандартів, пристроїв, способу їх зв’язку.

Метою дослідження є проведення аналізу, систематизації, порівняння протоколів передачі даних в IoT системах, огляд засобів керування «розумними речами» в мережах CAN та в цілому. Побудова моделі узгодження керування мережею розумних речей.

Об’єктом дослідження протоколи, що використовуються для побудови мереж розумних речей IoT систем.

Предметом дослідження є засоби та методи узгодження керування мережею IoT пристроїв.

1 АРХІТЕКТУРА IOT

Зрозумівши економічне значення IoT, важливо також розглянути технології, які використовуються в архітектурі IoT. Архітектурний шар IoT приймає форму, аналогічну стандартній моделі ISO (ISO) та IEC, Протоколу керування передачею (TCP) та Інтернет-протоколу (IP), а також 4-шаровій моделі Міністерства оборони США (DoD4)[1]. Таблиця 1.1 ілюструє вищезазначені моделі в межах архітектури інтерфейсу.

Насправді IoT архітектура могла бути реалізована на базі ISO, TCP та DoD4 без подальшого архітектурного моделювання, але вони не включають в себе особливості IoT, такі як зв'язок, збір та аналіз даних, керування пристроями, масштабованість, взаємосумісність, інтеграція та безпека. Таким чином, існувала потреба в реструктуризації всіх трьох моделей, щоб вони відповідали особливостям і проблемам IoT. Модель архітектури IoT складається з різних компонентів і є 4-шаровою центричною архітектурою, в якій на кожному рівні можна реалізувати конкретні технології. В таблиці 1.1 наведена 4-х шарова модель IoT

Таблиця 1.1 – Архітектурні шари

Архітектурні шари IoT	**Компоненти**
Прикладний рівень	Навколишнє середовище, енернетика, медицина, Транспорт
Рівень керування	Потоки даних, управління, безпека
Мережевий рівень	WiFi, Ethernet, Шлюзи
Рівень пристроїв	Мережі датчиків, актуатори, теги(RFID)

1.1 Рівень пристроїв

Низький шар в основному являє собою пристрої IoT, і вони містять різні типи та форми архітектури, властивості та можливості. Пристрій може

розглядатися як пристрій IoT, якщо він має будь-яку форму зв'язку, яку можна безпосередньо або опосередковано підключити до Інтернету. Пристрої на сенсорному рівні мають можливість розуміти та збирати інформацію в режимі реального часу для обробки. Вони мають низьку потужність та низьку швидкість передачі даних. Можна назвати такі області застосування деяких з цих датчиків: сенсори на тілі, датчики навколишнього середовища та датчики спостереження.

1.2 Мережевий рівень

Шлюз та мережевий рівень підтримує зв'язок пристроїв. Він складається з різноманітних протоколів, які допомагають у спілкуванні між пристроями та хмарою. Найбільш помітними з цих протоколів є протокол передачі гіпертексту (HTTP) з підходом RESTful, MQTT та протоколом обмежених програм (CoAP). Протоколи IoT будуть більш детально розглянуті в наступному розділі. Таблиця 1.2 показує шлюз та мережевий рівень із технологіями, які беруть участь у ньому.[2]

Таблиця 1.2 – IoT мережевий рівень

Компонент	**Різновид**
Мережа	WAN 3G, LTE, LTE-A, M LoRa, Sigfox, 5G, LAN WiFi, Ethernet
Шлюз	Мікроконтролери, міні-комп’ютери, модулі радіокомунікації, точки доступу

Крім того, одним із найважливіших аспектів шлюзу та мережевого рівня є його здатність збирати дані, а також бути посередником між пристроями. [3]. Шлюз та мережевий рівень також можуть підтримувати, наприклад, HTTP-сервер та брокер MQTT, щоб забезпечити зв'язок між пристроями. Крім того, він служить мостом і перетворює дані між різними протоколами, наприклад, збирати дані з датчиків по CAN та передавати їх через HTTP або MQTT.

1.3 Рівень управління

Рівень сервісу управління складається з двох основних функціональних частин, як зазначено в таблиці 1.1, - це частина моделювання пристрою, конфігурація та керування, а також частина управління потоком даних та контролю безпеки. Однак, перш ніж розглянути функції частин рівня керуючого сервісу, важливо також описати, що таке служби керування. У таблиці 1.3 наведено деякі служби, які може запропонувати рівень управління.[2]

Таблиця 1.3. – IoT рівень сервісів управління

Сервіси	**Компоненти сервісу**
Система єксплутаціонної підтримки	Управління пристроями, Конфігурація, Управління ефективність, Безпека
Аналітична платформа	Статистика, Data Mining, Прогнозування
Security	Контроль доступу, шифрування, ідентифікація
Business Rules Management (BRM)	Правила, моделювання, виконання
Business Process Management (BPM)	Моделювання потоків роботи

Як показано в таблиці 1.3, рівень управління сервісом виконує важливу роль в архітектурі IoT. Ролі можна згрупувати у дві частини. Управління службами даних відповідає за процеси, такі як інформаційна аналітика, контроль безпеки, моделювання процесів та управління пристроями. Управління даними має дві форми: періодичні та неперіодичні[2].

У періодичній обробці даних IoT інформація або дані періодично збираються датчиком IoT для аналізу. Наприклад, температурний датчик фіксує певну кількість інформації про погоду або стан промислової машини протягом певного проміжку часу. Проте не всі зібрані відомості будуть

необхідні для аналізу, отже, очищення даних, зібраних датчиком, необхідне для фільтрації.

У неперіодичній технології збору даних датчик IoT збирає дані і вимагає негайної реакції, як тільки відбувається деяка подія. Наприклад, якщо сенсорний пристрій IoT контролює пацієнта, доставка інформації повинна бути негайною і вимагає негайної відповіді. Існує також блок управління даними, який забезпечує управління потоком даних, доступу до інформації, інтеграції та контролю даних[2]. Існує також блок абстракції даних, який надає послуги, такі як обробка інформації.

1.4 Прикладний рівень

Прикладний рівень являє собою верхній шар, і це шар, який служить інтерфейсом між додатком датчика та кінцевими користувачами. Він складається з різних галузей застосування, таких як охорона навколишнього середовища, промисловість, охорона здоров'я, відстеження майна інтелектуальної власності та інші. Це також шар, на якому розміщуються такі протоколи, як HTTP, MQTT, CoAP, протокол чергування повідомлень (AMQP), протокол обміну повідомленнями XMPP, простий протокол доступу до об'єктів (SOAP). Перші три вищезгадані протоколи IoT будуть детально розглянуті в наступних розділах.

Крім того, оскільки різні програми з різних галузей мають різні протоколи та класифікації, виходячи з типу мережі, зони покриття, розміру, бізнес-моделі, систем реального часу, протоколи прикладного рівня можуть гарантувати зв'язок та обмін даними або інформацією між гетерогенними системами.

2 ПРОТОКОЛИ В IOT

2.1 Стандарти та протоколи фізичного рівня

2.1.1 Застосування WIFI в IoT

Wi-Fi - це стандартна бездротова мережа на основі специфікацій IEEE 802.11a/b/g/n. Протокол 802.11n забезпечує максимальну пропускну спроможність, але за рахунок високого енергоспоживання, таким чином пристрої IoT можуть використовувати лише 802.11b або g для збереження енергії. Незважаючи на те, що Wi-Fi використовується в багатьох прототипах та пристроях поточного покоління IoT, з’являються більш широко доступні рішення з великим діапазоном і низьким енергоспоживанням, які, швидше за все, витіснять Wi-Fi.[4]

Wi-Fi використовує високочастотні радіохвилі, а не дроти, щоб спілкуватися та передавати дані. Точка доступу об'єднує дротові та бездротові мережі та дозволяє надсилати та отримувати дані між бездротовими клієнтами та дротовою мережею. SSID, також відомий як "Назва мережі", являє собою ідентифікатор сервісного набору, який контролює доступ до певної бездротової мережі.

Переваги:

- Знімає мережеві пристрої з кабелів
- Дешевша розробка вбудованої системи
- Широка розповсюдженість
- Висока швидкість

Недоліки:

- 802.11b/g використовуйте спектр 2,4 ГГц, який переповнений іншими пристроями (Bluetooth , Zigbee)
- Споживання енергії
- Обмежений діапазон мережі
- Ризики безпеки (конфігурація)

2.1.2 Застосування Zigbee в IoT

ZigBee - це нова бездротова технологія. Технологічний стандарт, створений для мереж керування та датчиків. Заснована на стандарті IEEE 802.15.4, створеному Альянсом ZigBee. Такі компанії як Philips, Motorola, Intel, HP є членами альянсу[4]. В основному призначена для низького енергоспоживання, що, як правило, дозволяє батареям служити довше. ZigBee надає можливість створити повністю з'єднані будинки, де всі пристрої можуть спілкуватись, а також керуватись одним пристроєм. Це забезпечує мережеву безпеку. Специфікація ZigbEE 802.15.4 дозволяє створювати справжні mesh-мережі, але також підтримує такі топології як зірка та дерево.

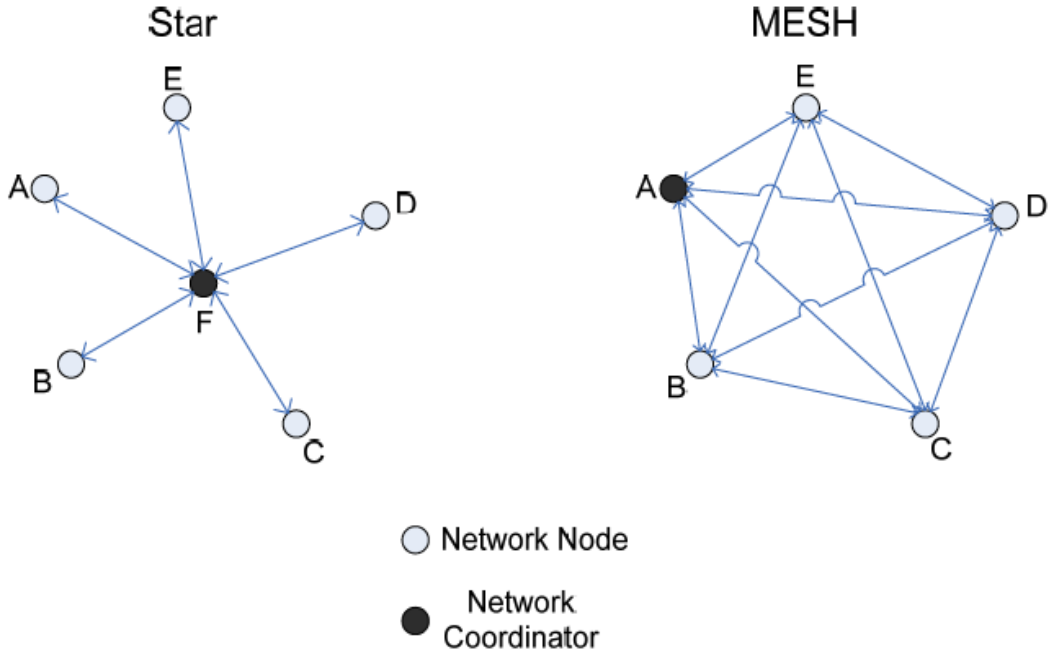


Рисунок 2.1 – Топології в ZigBee[4]

До переваг використання ZigBee можна віднести такі:

- Прямий зв'язок
- Більш легка розробка порівняно з Bluetooth.
- Низьке енергоспоживання
- Бездротовий зв'язок

Недоліками використання є:

- Робота на невеликій відстані з низькою швидкістю
- Потребує виплат за патент, а тому дорогий у впроваджені

Застосування Zigbee:

- Побутова електроніка
- Охорона здоров'я
- Комерційний та житловий контроль
- Промислові та державні ринки у всьому світі
- Домашні мережі
- Промислове управління

2.1.3 Застосування Bluetooth в IoT

Це відкритий стандарт для розвитку персональної мережі (PAN). Його функції включають низьку вартість, низьку потужність та короткий діапазон, як правило, близько 10 метрів. Може обмінюватися інформацією з іншими пристроями Bluetooth. Використовується для створення невеликої мережі пристроїв, близьких один до одного. Підтримує стадарт IEEE802.15.4 і працює в частотному діапазоні 2,4 ГГц.

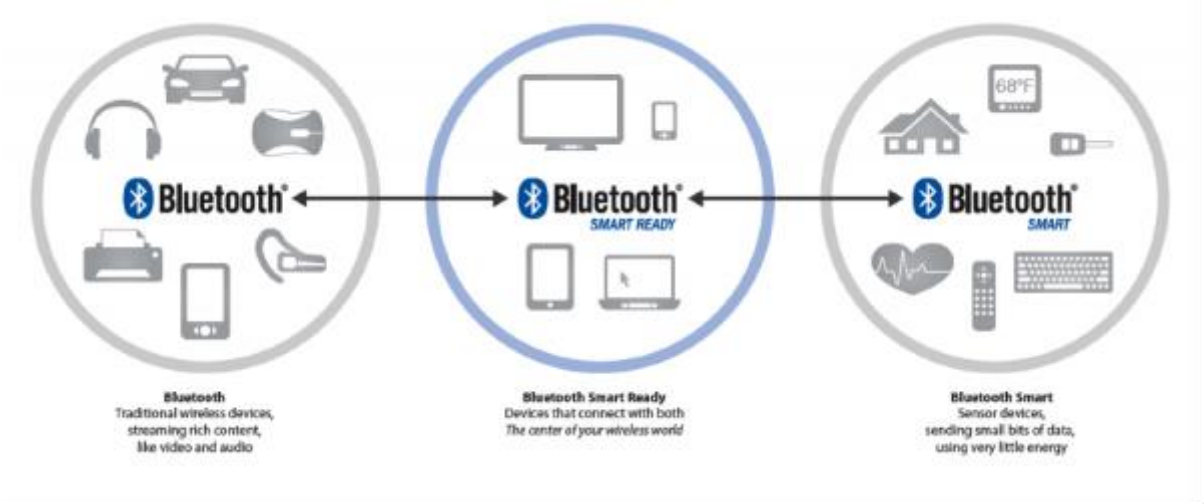


Рисунок 2.2 – Технології Bluetooth[4]

Технологія Bluetooth поділяється на три типи, як показано на рисунку 2.2:

- Bluetooth Classic – традиційний Bluetooth має більшу пропускну здатність, зазвичай використовується для передачі аудіо та файлів.
- Bluetooth Smart – Bluetooth Low Energy під брендом «Bluetooth Smart» використовується для передачі невеликих обсягів інформації. Розроблений для додатків з невеликим циклом постачання даних. Bluetooth Smart не можуть з'єднуватись з Bluetooth Classic пристроями.
- Bluetooth SmartReady – це хаб пристрої, які підтримують з'єднання як Bluetooth Classic так і з Bluetooth Smart.

Розглянемо детальніше BLE. Це малопотужний варіант популярного протоколу бездротового зв'язку Bluetooth. Він призначений для зв'язку на малих відстанях (не більше 100 м), як правило, у топології зірка, з одним основним пристроєм, який керує кількома додатковими пристроями. Bluetooth працює як на шарах 1 (PHY), так і на 2 (MAC) моделі OSI. BLE найкраще підходить для пристроїв, які передають невеликі обсяги даних у надриви, оскільки пристрої призначені для спання влада, коли вони не передають дані. Особисті пристрої IoT, такі як носимі спортивні прилади, часто використовують BLE.

2.1.4 Застосування 6LOWPAN в IoT

Стандарт 6LoWPAN дозволяє підключити багато пристроїв до хмари. Мала потужність, IP-адресні вузли та підтримка більших мережевих мереж роблять цю бездротову технологію гарним вибором для Інтернету речей. Повна назва «IPv6 через бездротові локальні бездротові мережі» означає «передачу пакетів IPv6 через малопотужні бездротові персональні мереж», отже, 6LoWPAN є мережевою технологією або адаптаційним рівнем, що дозволяє ефективно передавати IPv6 пакети в невеликих фреймах канального рівня, визначених в безпровідному стандарт IEEE 802.15.4. Побудова наскрізної інфраструктури на базі Інтернет-протоколів (IP) вбрало в себе всі досягнення більш ніж 30-річного розвитку технології IP, це сприяло відкритим

стандартам та їх сумісності, що в значній мірі демонструвалося щоденним використанням Інтернету його майже 3 мільярдами користувачів.

6LoWPAN - відкритий стандарт, визначений в RFC 6282 «Робочою група з розробки Інтернету» (IETF) - організацією по стандартизації, яка визначила багато відкритих стандартів, що використовуються в Інтернеті, такі як UDP, TCP і HTTP. Чудова особливість 6LoWPAN полягає в тому, що він був спочатку задуманий, щоб підтримувати малопотужні бездротові мережі 2,4 ГГц, побудовані на базі IEEE 802.15.4, але тепер цей стандарт адаптований і використовується в безлічі інших фізичних мережах передач, включаючи бездротові мережі в діапазонах нижче 1 ГГц, Bluetooth Smart і мережі Wi-Fi.

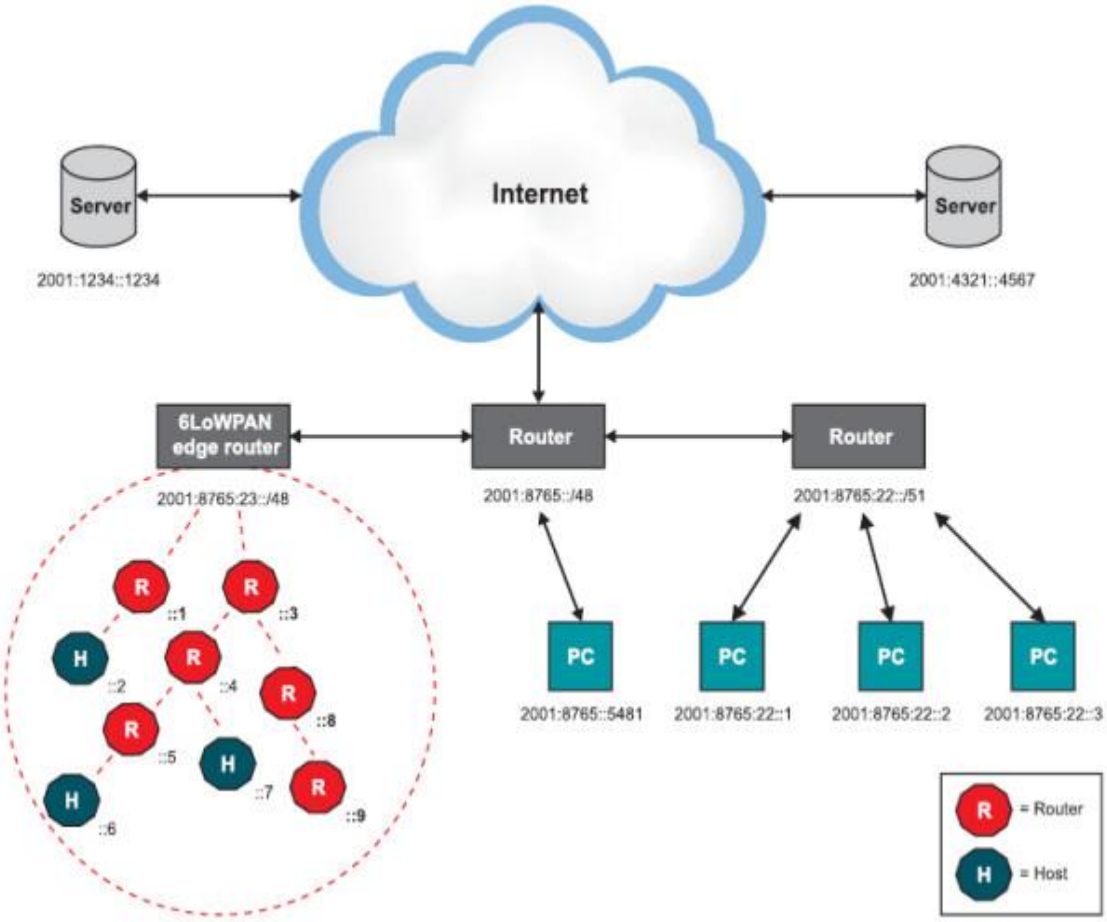


Рисунок 2.3 – Приклад мережі 6LoWPAN[6]

На рисунку 2.3 зображено приклад мережі IPv6, що включає мережу 6LoWPAN. Вихідний канал до Інтернет забезпечується точкою доступу (AP), що діє як IPv6 маршрутизатор. У типовій конфігурації до точки доступу

підключається декілька різних пристроїв, таких як ПК, сервери і так далі. 6LoWPAN-мережа пов'язана з IPv6-мережею за допомогою використання граничного маршрутизатора. Граничний маршрутизатор виконує три функції: обмін даними між пристроями 6LoWPAN та Інтернетом (або іншою IPv6-мережею), локальний обмін даними між пристроями в 6LoWPAN-мережі та формування й обслуговування радіопідмережі (6LoWPAN-мережі).

2.1.5 Застосування CAN в IoT

Шина CAN - це протокол, що базується на повідомленнях, і дозволяє індивідуальним системам, пристроям і контролерам спілкуватися у межах мережі.

Після його введення в середині 1980-х років, шина CAN розвивалася далеко за межами автомобільної промисловості, де проткол був вперше широко прийнятий. До того як шина CAN отримала популярність, джгут проводів транспортного засобу міг містити значну довжину проводу, з пучками 8 або більше проводів, необхідних для транспортування різних сигналів до і з взаємозв'язаних систем автомобіля. На відміну від них, CAN шина використовує високошвидкісну (25 кбіт/с - 1 Мбіт/с) систему витої пари, що значно зменшує кількість дроту, необхідного для забезпечення ефективної комунікації компонентів системи.

Комунікаційна шина CAN забезпечує ряд переваг для користувачів, зокрема:

- Швидкість передачі даних - швидкість передавання даних CAN далеко випереджає традиційні аналогові провідні жорсткі кабелі, оскільки одночасно декілька повідомлень можуть надсилатися всім підключеним пристроям, датчикам або виконавчим елементам.

- Гнучкість - завдяки тому, що він обладнаний двома жилами, одноканальний, двошаровий, шина CAN пропонує покращену гнучкість встановлення та обслуговування. CAN-підключені системи не тільки містять значно менше дроту, що робить їх простішим для встановлення, але додавання нових компонентів до системи вимагає набагато меншого розвитку, а також значно зменшує ускладнення при діагностиці та вирішенні проблем із сигналом.
- Надійність. На додаток до того, що він набагато менш чутливий до магнітних перешкод, ніж аналогові, CAN-зв'язок також потребує менше кабелів і роз'ємів, різко зменшуючи точки відмови.
- Вартість. Нижчі витрати на апаратне забезпечення та мінімальні вимоги щодо обробки сигналів робить CAN ідеальним рішенням для вбудованих систем.

CAN - одна з найбільш надійних мережевих технологій, особливо якщо використовувати лінійну топологію з дуже короткими заглушками та виту пару кабелю. Перехрещена пара мідних кабелів із звичайною землею зазвичай реалізує фізичну передачу. Звичайно, всі пов'язані вузли повинні підтримувати однакову швидкість передачі даних і ті ж самі параметри бітової синхронізації. Найпоширенішою є високошвидкісна передача, стандартизована в ISO 11898-2: 2003. Він підтримує швидкість передачі даних до 1 Мбіт / с. Існують також високошвидкісні передавачі з малою потужністю (ISO 11898-5) та вибіркові функції пробудження для часткових мереж (ISO 11898-6). Всі три стандарти, згадані вище, об'єднані в один стандарт (ISO 11898-2: 2016).

Потужність високошвидкісних CAN-мереж відмінна. В автотранспортних мережах часто використовуються топології зірки, іноді з кількома зірками. У деяких програмах використовуються гібридні топології, що поєднують лінію та зірку. Проте найпотужнішою топологією є шина з дуже короткими заглушками.

Інший рівень фізичного рівня, який використовується в промисловості, складається з ISO 11898-3, так званої "відмовостійкої", малої потужності передачі. Він не широко використовується в інших галузях промисловості, і обмежується 125 кбіт / с. Фізичний рівень, специфічний для програми, стандартизований в ISO 11992-1, який також обмежується 125 кбіт / с. Вона використовується тільки в європейських мережах "точка-точка" для вантажних автомобілів-причепів. Однопровідна передача, як зазначено в SAE J2411, має деякі обмеження стосовно надійності та швидкості передачі.

Оскільки Інтернет речей продовжує зростати та охоплювати все більш складні системи, стандартизація способів зв'язку кожного компонента з наступним буде важливою для забезпечення сумісності, розширюваності та довговічності встановлення. Таким чином CAN шину зв'язку в широкому можна застосовувати в щирокому діапазоні користувальницьких вбудованих систем, особливо в середовищах виробництва та в сучасних установках автоматизації будівель.

2.2 Протоколи прикладного рівня

2.2.1 Обмін даними за допомогою MQTT

IBM винайшов телеметричний транспортний переказ повідомлень (MQTT) для супутникового зв'язку з нафтовим обладнання. Має високу надійність і низьку потужність, і тому було доцільно застосовувати в мережах IoT. Стандарт MQTT з тих пір був прийнятий суспільством відкритих стандартів OASIS і випущений під версією 3.1.1. Він підтримується в спільноті Eclipse, а також багатьма комерційними компаніями, які пропонують пакет з відкритим кодом та консалтинг.[7]

MQTT використовує модель "публікування/підписки" (Publish/Subscribe), і вимагає наявності центрального брокера MQTT для керування та маршрутизації повідомлень серед вузлів мережі MQTT. Eclipse

описує MQTT як "комунікаційний протокол між багатьма користувачами для передачі повідомлень між декількома клієнтами через центральний брокер".

MQTT використовує TCP для його транспортного рівня.

Модель "Publish / Subscribe" MQTT чудово масштабується. Брокери та вузли публікують інформацію, а інші підписуються відповідно до змісту, типу чи предмета повідомлення. (Це стандартні умови MQTT.) Як правило, брокер підписується на всі повідомлення, а потім маршрутизує інформацію до своїх вузлів. Є кілька конкретних переваг для Pub/Sub моделі. Поки вузол та брокер повинні мати IP-адреси один одного, вузли можуть публікувати інформацію та підписатись на опубліковану інформацію інших вузлів без будь-яких знань один одного, оскільки всі дані проходять через центральний брокер. Це зменшує накладні витрати, які можуть супроводжувати TCP-сеанси, і дозволяє кінцевим вузлам працювати незалежно один від одного. Вузол може публікувати свою інформацію незалежно від стану інших вузлів. Інші вузли можуть одержувати опубліковані відомості від брокера, коли вони активні. Це дозволяє вузлам залишатися в сонних станах, навіть коли інші вузли публікують повідомлення, які мають безпосереднє відношення до них.

MQTT використовує незашифрований TCP і не захищений поза межами коду. Але оскільки він використовує TCP, він може і повинен - використовувати інтерфейс безпеки TLS/SSL. TLS - це безпечний спосіб шифрування трафіку, але також ресурсоємний для легких клієнтів через необхідність в рукостисканні та підвищенню надлишкових даних в пакеті. Для мереж, де енергоспоживання є дуже високим пріоритетом та безпека набагато меншою, цілком достатньо шифрування корисної завантаженості пакетів.

Термін "QoS" має інше значення в рамках MQTT. У MQTT рівні "0", "1" та "QoS" описують підвищення рівнів гарантованої передачі повідомлень.

MQTT Рівень 0 (Хоча б один раз). З цим рівнем QoS повідомлення доставляються відповідно до гарантій доставки основної мережі TCP/IP.

MQTT QoS Level 1 (принаймні один раз). З цією семантикою доставки повідомлення гарантовано надходять на сервер і повинні бути підтверджені

(PUBACK). Проте, може бути Duplicate (DUP), який виникнає через затримку приходу підтвердження (PUBACK) або відмови будь-якої лінії зв'язку або відправника. Це означає, що коли відправник (клієнт) видає дані, після деякого часу, якщо PUBACK не буде отримано, він знову повторно надсилає дані з бітом DUP, встановленим у заголовку повідомлення, в результаті копіювання повідомлень. Проте додаток може відхилити дублікат повідомлення, оцінивши поле ідентифікатора повідомлення. Сценарій застосування може бути датчик, який контролює стан дверей. Це означає, що стан дверей - OPEN / CLOSE або CLOSE / OPEN, і ці зміни станів публікуються для абонентів, наприклад, у вигляді сигналу. На рисунку 2.4 нижче показана семантика доставки QoS рівня 1

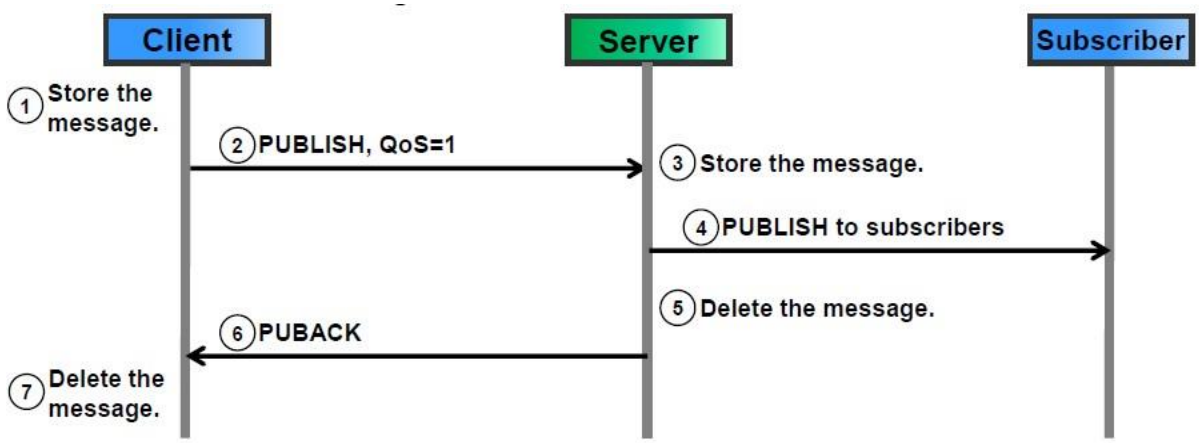


Рисунок 2.4 – Доставка повідомлення в QoS Level 1 [9]

MQTT QoS Level 2 (точно один раз)

Це найвищий, безпечний, але повільний рівень QoS, і він гарантує, що кожне повідомлення одержує лише один раз за підпискою. Має більшу частину накладних витрат з точки зору контрольних повідомлень та необхідності локального зберігання повідомлень. Це також комбінація QoS Level 0 та QoS Level 1.

З QoS рівня 2, коли одержувач отримав повідомлення PUBLISH, він обробляє повідомлення та підтверджує його повідомленням PUBREC. Приймач також зберігає повідомлення з посиланням на ідентифікатор повідомлення, доки він не відправив PUBCOMP. Це потрібно, щоб уникнути

дублювання обробки повідомлення. Крім того, повідомлення PUBLISH, що зберігається у клієнта, може бути відкинуте після отримання PUBREC. Повідомлення PUBREC зберігається після прибуття, а клієнт відповідає PUBREL. Приймач з іншого боку також видаляє всі збережені повідомлення після отримання PUBREL, і подібна подія відбувається на стороні клієнта після отримання PUBCOMP від абонента. На рисунку 2.15 пояснюється семантика QoS рівня 2.

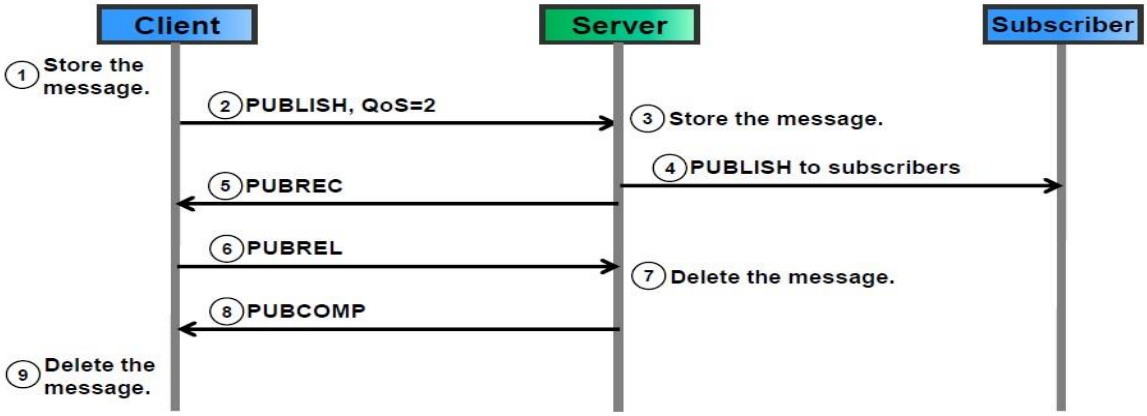


Рисунок 2.5 – Доставка повідомлення в QoS Level 2 [9]

MQTT забезпечує повідомлення "останнього волі та заповіту (LWT)", яке можна зберегти в брокері MQTT, якщо вузол несподівано відключився від мережі. LWT повідомлення зберігає стан і цілі вузла, включаючи типи команд, які він опублікував, та його підписки. Якщо вузол зникає, брокер повідомляє всіх абонентів LWT вузла. І якщо вузол повертається, брокер повідомляє про його попередній стан. Ця функція чудово поєднує слабкі мережі та масштабованість.

У системах, які вже мають центральний брокер, він може стати єдиною точкою відмови для всієї мережі. Наприклад, якщо брокер - це вузол із джерелом живлення, який не підтримує роботу від акумулятора, то вузли, що працюють на акумуляторі, можуть продовжувати працювати під час відключення електроенергії, коли брокер також відключений, таким чином, мережа не працює.

2.2.2 Обмін даними за допомогою COAP

З зростаючою важливістю IoT, «Робоча група з розробки Інтернету» (IETF) взяла легкий обмін повідомленнями та визначила протокол обмеженого застосування - CoAP. Як визначено в IETF, CoAP призначений для "використання з обмеженими вузлами в обмежених мережах(наприклад, з малою потужністю, втратами)". Співтовариство Eclipse також підтримує CoAP як відкритий стандарт, так само як і MQTT CoAP комерційно підтримується та швидко розвивається провайдерами IoT.

CoAP - це клієнт-серверний протокол, що забезпечує інтерактивну взаємодію моделі "запит/звіт". На відміну від MQTT, який був адаптований до потреб IoT за допомогою десятирічного розвитку, IETF визначив CoAP з самого початку для підтримки IoT з легкими повідомленнями для пристроїв, що працюють в обмеженому середовищі. CoAP призначений для взаємодії з HTTP і RESTful web через прості проксі, що робить його з самого початку сумісним з Інтернетом.

CoAP працює через UDP, який за своєю суттю менш надійний, ніж TCP. Наприклад, датчик температури може надсилати оновлення кожні кілька секунд, навіть якщо ніщо не відбувалось змін температури. Якщо вузол, що пропускає одне оновлення, наступне прибуде через кілька секунд і, ймовірно, не сильно відрізнятиметься від попереднього.

Датаграми UDP також дозволяють швидше пробуджувати пристрої та передавати пакети з меншими накладними витратами. Це дозволяє пристроям залишатися у сонному стані протягом тривалого періоду часу, зберігаючи енергію акумулятора.

Мережа CoAP, по суті, один до одного; однак можлива поєднання один до багатьох, або багато до багатьох, оскільки протокол побудовано поверх IPv6. Важливо відмітити, що передача багатоадресних повідомлень на пристрої, які знаходяться у стані сну, є ненадійною або може вплинути на

тривалість роботи акумулятора, якщо вони повинні регулярно прокидатися, щоб отримувати ці повідомлення.

CoAP використовує DTLS для безпеки у верхній частині свого транспортного протоколу UDP. Подібно до TCP, UDP не шифрується, але може бути доповнений DTLS.

CoAP використовує URI, щоб забезпечити стандартні очікування презентації та взаємодії для мережевих вузлів. Це дає певну ступінь автономії в пакетах повідомлень, оскільки можливості цільового вузла частково описуються деталями його URI. Іншими словами, вузол сенсора, що працює від акумулятора, може мати один тип URI, тоді як пристрій керування потоком живлення може мати інший. Вузли, що з'єднуються з вузлом датчика живлення від акумулятора, можуть бути запрограмовані на очікування довшого часу відгуку, повторюваної інформації та обмежених типів повідомлень.

В рамках протоколу CoAP більшість повідомлень надсилаються та отримуються за допомогою моделі запиту / звіту; однак існують і інші режими роботи, які дозволяють дещо відокремити вузли. Наприклад, CoAP має спрощений механізм "спостерігати", подібний до Pub/Sub MQTT, що дозволяє вузлам спостерігати за іншими.

В даний час існують проекти доповнень до стандарту, які можуть надати CoAP аналогічну функцію до Pub/Sub моделей MQTT в короткостроковій та середньостроковій перспективі. Провідним кандидатом на сьогодні є проект пропозиції, який дозволив мережам CoAP реалізовувати таку Pub/Sub модель, як згадане вище у MQTT.

MQTT в даний час є більш зрілим і стабільним стандартом, ніж CoAP. Тим не менш, CoAP має величезний ринковий імпульс і швидко розвивається, щоб забезпечити стандартизований фундамент з важливими доповненнями.

Цілком імовірно, що в найближчому майбутньому CoAP досягне подібного рівня стабільності та зрілості як MQTT. Але стандарт зараз розвивається, що може призвести до деяких проблем із сумісністю.

Надійність CoAP - це аналог в QoS MQTT. Протокол надає дуже простий спосіб надання "підтвердженого" повідомлення та "не підтвердженого" повідомлення. Підтвердження повідомлення гарантується за допомогою ACK від передбаченого одержувача. Непідтвердженим повідомленням є методологія "Fire and Forget".

2.2.3 Обмін даними за допомогою HTTP/REST

HTTP - це найбільш застовований і адаптований протокол прикладного рівня у World Wide Web. Стандартизація HTTP була виконана Робочою групою з розробки Інтернету (IETF) у співпраці з консорціумом World Wide Web (W3C) . HTTP працює на технології обміну повідомленнями клієнта-сервера, в якій клієнт запитує сторінку з гіпертекстової розмітки (HTML) з сервера, який надає відповідь з HTML-сторінкою. HTTP покладається на TCP як транспортний протокол, який використовує сокети для передачі даних. Зв'язок між клієнтом і сервером починається з клієнта через сокет з'єднання на порту 80 (стандартний порт обраний косорціумом, звісно можливе використання будь-якого іншого), який є призначеним номером порту для HTTP сервера. Коли встановлено з'єднання, сервер приймає запит клієнта в форматі HTML-сторінки, та інших об'єктів [8].Проте, після встановлення з'єднання, HTML-сторінки та об'єкти потім передаються між клієнтським браузером та веб-сервером. Після завершення запиту TCP припиняє зв'язок між клієнтом і сервером, а також очищує пам'ять, щоб видалити попередні запити від клієнта. В HTTP такі запити як GET, PUT, POST та DELETE, є чотирма основними методами. Запит GET відображає веб-сторінку та її об'єкти. Методи запиту PUT і POST використовуються для зміни серверних ресурсів, а запит DELETE видаляє ресурси, які не потрібні.

Крім того, існує два типи HTTP-з'єднання, які можна встановити за допомогою TCP. Це нестійкі (HTTP/1.0) та постійні (HTTP/1.1) з'єднання. Основна відмінність нестаціонарних (HTTP/1.0) і стійких (HTTP/1.1) залежить

від кількості TCP-з'єднань, необхідних для передачі уніфікованого локатора ресурсів (URL-адреси) веб-сторінки та її об'єктів. На рисунку 4 показана схема HTTP з'єднання між клієнтом і сервером.

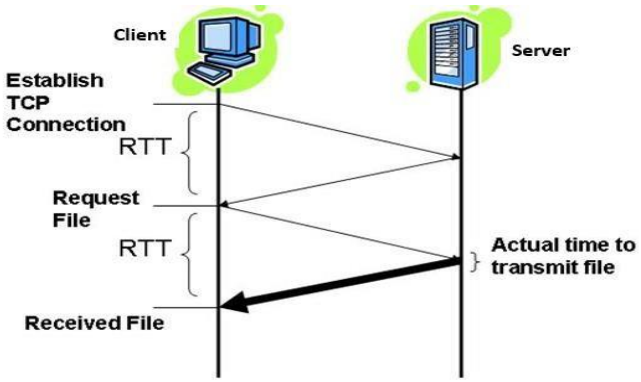


Рисунок 2.6 HTTP-з'єднання TCP-з'єднання між клієнтом і сервером[9]

На рисунку 2.6 RTT – це час, витрачений на відправку пакета з клієнта на сервер та отримання відповіді. Він також відображає час, необхідний для встановлення TCP-з'єднання, надсилання запиту та отримання відповіді або отримання файлу з його часом передачі. Математично загальна RTT, починаючи з початку створення TCP-зв'язку до отримання запитуваного файлу, може бути виражена як 2 x RTT + час передачі файлів (FTT).

Основною перевагою використання REST, як з точки зору клієнта, так і з точки зору сервера, є взаємодії на основі REST, використовуючи конструкції, добре знайомі кожному, хто звик до використання протоколу HTTP

Прикладом цієї схеми є взаємодії на основі REST, які повідомляють про свій статус за допомогою стандартних кодів статусу HTTP. Отже, 404 означає, що запитуваний ресурс не знайдено; код 401 означає, що запит не був дозволений; код 200 означає, що все нормально; і код 500 означає, що на сервері виникла невиправна помилка програми.

REST вимагає хоча б один URI для доступу до кожного ресурсу. RESTful використовують ієрархію подібну до списку директорій. URI не каже нічого

про тип операції, він визначається методом HTTP. Наприклад, щоб отримати деяку інформацію про користувача, використовується наступна адреса:

http://service/user/1

В Rest архітектурі, система повинна мати єдиний інтерфейс. HTTP надає список дій як показано в таблиці 2.1

Таблиця 2.1 – Інтерфейси доступу до ресурсів

Метод	**Операція на сервері**	**Тип**
GET	Читання ресурсів	Безпечний
POST	Додати або оновити ресурс	Ідемпотентний
PUT	Додати або оновити ресурс	Ідемпотентний
OPTIONS	Список дозволених операцій	Безпечний
DELETE	Видалити ресурс	Ідемпотентний

Безпечною називається операція, яка не має ніякого впливу на оригінальне значення ресурсу. Ідемпотентна операція завжди повертає одинаковий результат, незалежно від того скільки разів було виконана.

REST не зберігає стани для всіх клієнтів на сервері. Запит не може покладатися на минулий, а тому сервіси обробляють кожний новий запит незалежно. Кешування – це концепт збереження згенерованих результатів і подальше використання збережених результатів, для пришвидшення відповіді на запит та покращення продуктивності сервісів, але при неправильному використанні може призвести до того, що клієнти будуть отримувати застарілі результати.

Такі властивості як шифрування та цілісність передачі даних, вирішуються не шляхом додавання нових фреймворків чи технологій, а

покладаються на відомі засоби шифрування протоколу SSL та Security Layer Security (TLS). Отже, вся архітектура REST побудована на концепціях, з якими більшість розробників вже знайомі.

REST також є незалежним від мови архітектурним стилем. Програми на основі REST можуть бути написані будь-якою мовою, наприклад Python, Java, C++ або JavaScript. Поки за допомогою мови програмування можна створювати веб-запити HTTP, ця мова може бути використана для виклику RESTful API або веб-служби. Подібним чином, RESTful веб-сервіси можуть бути написані на будь-якій мові, тому розробники, яким потрібно виконувати такі послуги, можуть вибирати технології, які найкраще підходять для їх ситуації. Іншою перевагою використання REST є його проникливість.

Проте, використання REST за допомогою HTTP-конструктів також створює обмеження. Багато обмежень HTTP також стають недоліками цього архітектурного стилю. Так само, оскільки HTTP не має жодного механізму для відправки push-повідомлень від сервера до клієнта, важко реалізувати будь-які типи послуг, на яких сервер оновлює клієнт без використання клієнтського опитування сервера.

З точки зору впровадження, загальною проблемою REST є той факт, що розробники не погоджуються з тим, що означає бути REST. Деякі розробники програмного забезпечення неправильно вважають все, що не є SOAP-базою, як RESTful. Але це загально невірне уявлення про REST, той факт, що це архітектурний стиль означає, що немає еталонної реалізації або остаточного стандарту, який підтвердить, чи є даний дизайн RESTful. Як наслідок, існує дискусія щодо того, чи відповідає деякий API принципам на основі REST.

2.3 Висновки

Вибір технологій мережі в IoT являє собою компроміс в цілому. Так як буде впливати на дизайн пристроїв, існують залежності між великою кількістю параметрів. Наприклад, діапазон мережі, швидкість передачі даних та споживання енергії пов'язані між собою. Якщо збільшити діапазон мережі або швидкість та обсяг переданих даних, пристрої IoT, найімовірніше, вимагатимуть додаткової потужності для передачі даних в цих умовах.

Проаналізовані фізичні протоколи надають різноманітні можливості, але варто виділити серед них CAN. Висока швидкість в рамках датчиків, надійність, чудова можливість коригування помилок, надають перевагу в таких сферах як промисловість, медицина, автоматизація, де існують високі вимоги щодо якості передачі даних.

3 ОГЛЯД IOT ПЛАТФОРМ

У найпростішій формі платформа IoT забезпечує зв'язок між "речами" або пристроями. Архітектура може також складатися з програмної платформи, платформи розробки додатків або платформи аналітики.

Існує чотири типи платформ, які часто називають платформою IoT[11]:

- Connectivity/M2M платформами. Ці платформи зосереджені в основному на підключенні з'єднаних пристроїв IoT через телекомунікаційні мережі, але рідко на обробку різних наборів даних з датчиків. (Прикладом платформи підключення є Sierra Wireless 'AirVantage)
- IaaS backends. Інфраструктура-як-сервіс - сервери, що надають хостинг-простір і обчислювальні потужності для додатків і сервісів, раніше оптимізували для десктопів і мобільних додатків, але зараз в фокус потрапив і IoT (приклад - IBM Bluemix, але не IBM IoT Foundation).
- Технічні платформи для конкретного програмного забезпечення. Деякі компанії, що продають підключені пристрої, створили свій власний програмний пакет. Вони звертаються до бекенда як платформи IoT. Оскільки платформа не відкрита для кого-небудь на ринку, це дискусійно, чи слід назвати це платформою IoT. (Прикладом є Google Nest)
- Розширення програмного забезпечення для споживача / підприємства. Існуючі корпоративні програмні пакети та операційні системи, такі як Microsoft Windows 10, все більше дозволяють інтегрувати пристрої IoT. В даний час ці розширення часто недостатньо просунуті, щоб класифікувати як повну платформу IoT.

У більш витонченому вигляді справжня платформа IoT складається з восьми важливих архітектурних блоків зображених на рис. 3.1:

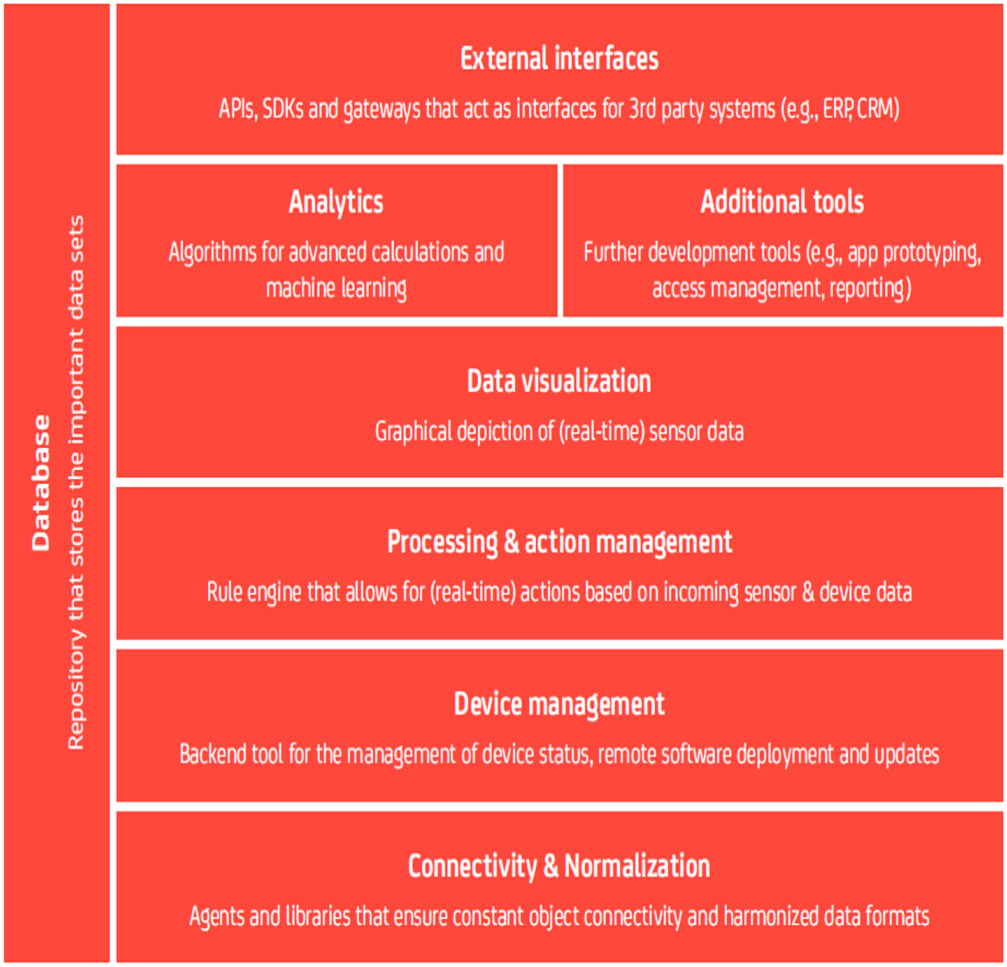


Рисунок 3.1 – Архітектура IoT платформ [11]

3.1 Хмарна платформа Google Cloud Platform IOT

Google Cloud Platform підтримує сценарії IoT для пристроїв у хмарі за допомогою Cloud Pub/Sub. Cloud Pub/Sub підтримує широкомасштабне надсилання повідомлень через протокол HTTP (REST) або gRPC - компактний формат обміну повідомленнями, що використовує протоколи буферів. Клієнтські бібліотеки існують для Go, Java (Android), .NET, JavaScript, Objective-C (iOS), PHP, Python і Ruby. Ціноутворення залежить від кількості операцій та витрат на зберігання (це, як правило, мінімальне при відсутності невдалих поставок) плюс витрати мережі, якщо споживач знаходиться в іншому регіоні. Обробка типового повідомлення включатиме 3 операції, 1 публікація, 1 обробка та 1 підтвердження. Повідомлення поділяються на блоки в 64 Кбайт, кожен з яких вважається повідомленням для цілей розрахунків.

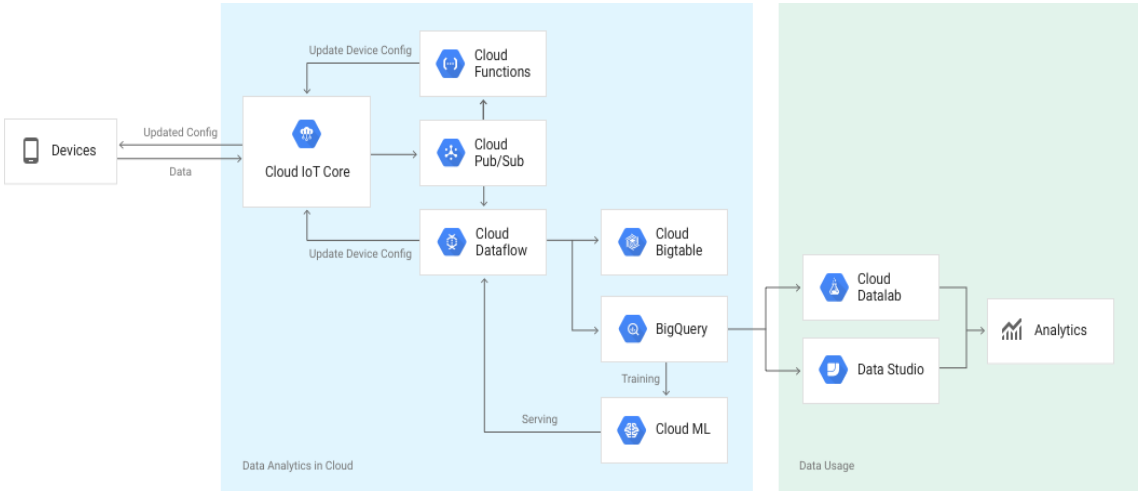


Рисунок 3.2 – Google Cloud Platform IOT

Коли повідомлення надходять, Dataflow може бути використаний для обробки вхідного потоку, наприклад, переміщення даних на зберігання та виконання аналізу потоку в реальному часі. Дані також можуть бути передані у великий запит, рішення для зберігання даних від Google за допомогою API потоку великих запитів. Також легко виконати спеціальну логіку над окремими повідомленнями, коли вони надходять через Google Cloud Functions.

Firebase - власна мобільна платформа Google і розробка для IoT. Firebase підтримує обмін повідомленнями між пристроями в хмарі через HTTP і XMPP, а також SDK для iOS, C ++, JavaScript та, звичайно, Android. Він поставляється з власною базою NoSQL, яка буде автоматично синхронізувати стан пристрою, включаючи фотографії, відео та зображення. Є також багата вбудоване журналювання та підтримка моніторингу.

Компанія Firebase була придбана компанією Google у 2014 році і пропонується як окремий продукт, але також добре інтегрується з хмарними сервісами Google, доступний комбінований білінг. Ця послуга доступна на безкоштовному рівні, передбачуваний варіант оплати за місяць або варіант оплати за покупки (ціна залежить від використовуваного сховища та послуг).

Подібно до Microsoft, у Google є своя операційна система IoT на базі Android, відома як Brillo. Brillo оснащений вбудованою підтримкою Weave, нової комунікаційної платформи, розробленої спеціально для того, щоб дозволити взаємодіяти пристроям IoT та контролерам. Оскільки більшість

пристроїв IoT контролюються телефоном, планшетом або подібним пристроєм, Weave забезпечує стандартний підхід, що дозволяє взаємодіяти між собою телефонам, пристроям IoT та хмарам. Google сподівається, що Weave стане стандартом для всіх комунікацій IoT, що в кінцевому підсумку призведе до розумніших рішень IoT.

3.2 Хмарна платформа Azure IoT Hub

IoT Hub - Azure IoT рішення для «розумних пристроїв» для комунікації з хмарними технологіями. IoT Hub підтримує AMQP, MQTT та HTTP. Якщо пристрій не підтримує один з цих протоколів, можна адаптувати вхідний і вихідний трафік за допомогою шлюзу Azure IoT Protocol Gateway. Набір SDK пристроїв доступний для NET, JavaScript, Java, C і Python. IoT Hub забезпечує реєстр пристроїв, який підтримує список пристроїв і забезпечує доступ до певної черги пристрою для надійної зв'язку з певним пристроєм. Отримані дані можуть бути відправлені в Blob Storage для обробки архівів або офлайнової інформації, або відправлені до кінцевої точки концентратора подій для негайної обробки. Існує також підтримка моніторингу та діагностики IT. IoT-концентратор поставляється в 4-х рівнях, починаючи від безкоштовного рівня до високого рівня продуктивності S3, який може підтримувати до 300 000 000 повідомлень на день. Додаткові одиниці можна придбати для кожного рівня для збільшення пропускної спроможності, якщо це потрібно. Повідомлення надсилаються в 4 Кб блоку.

Центри подій - це ще один варіант для сценаріїв "пристрій до хмари", і може бути кращим рішенням для прийому даних від пристроїв у великих масштабах. Центри подій можуть приймати великі обсяги повідомлень через AMQP та HTTP. Продуктивність концентраторів подій вимірюється в одиницях пропускної спроможності (TU), де кожний TU дозволяє отримати 1 Мб / S, хоча є можливість збільшити, якщо звернутись до служби підтримки. Ціноутворення базується на кількості подій (на мільйон) та сплаті за кожну пропускну одиницю за годину.

Центри подій часто використовуються в Azure Stream Analytics для аналізу даних пристрою в реальному часі. Він використовує SQL-подібну мову для виконання запитів по вхідному потоку даних і може обробити дані за допомогою інтеграції інших служб Azure, таких як Azure Machine Learning. Stream Analytics може виводитись на більшість рішень для зберігання даних в Azure або безпосередньо на Power BI для візуалізацій, центри подій.

Також можна передавати дані в Apache Storm - популярну платформу аналізу потокового потоку з відкритим вихідним кодом.

Microsoft, напевно, уважно стежите за тим, щоб Azure IoT використовувався з будь-яким пристроєм. З оголошеною ядром Windows 10 IoT, версією Windows 10, розробленою спеціально для роботи на пристроях IoT, Microsoft також позиціонує себе як ціле рішення IoT-провайдера.

3.3 Хмарна платформа AWS

AWS IoT - це повністю керована платформа для побудови повних рішень IoT на AWS. Пристрої спілкуються з додатками, що працюють у хмарі через HTTP, MQTT та WebSockets, які захищені за допомогою TLS. Спеціальні SDK для пристроїв доступні для Embedded C, JavaScript, Python, iOS, Android та Arduino Yún. AWS IoT підтримує надійні сценарії обміну повідомленнями від Cloud-to-Device та Device-to-Cloud, навіть якщо пристрій не підключено. Ціноутворення є відносно прямою, плата за 1 млн. повідомлень, надісланих або отриманих (доступний також безкоштовний рівень низької пропускної спроможності). Повідомлення обробляються блоках, розміром 512 байт, максимально 128 Кб.

IoT поставляється з движком декларативних правил, який використовується для перетворення та маршруту трафіку IoT до певного місця або кінцевої точки, наприклад, S3 або функції лямбда. Також можна спрямувати дані до Streaming Kinesis, які можуть запускати аналітику в реальному часі за допомогою програм, написаних за допомогою клієнтської бібліотеки Kenisis.

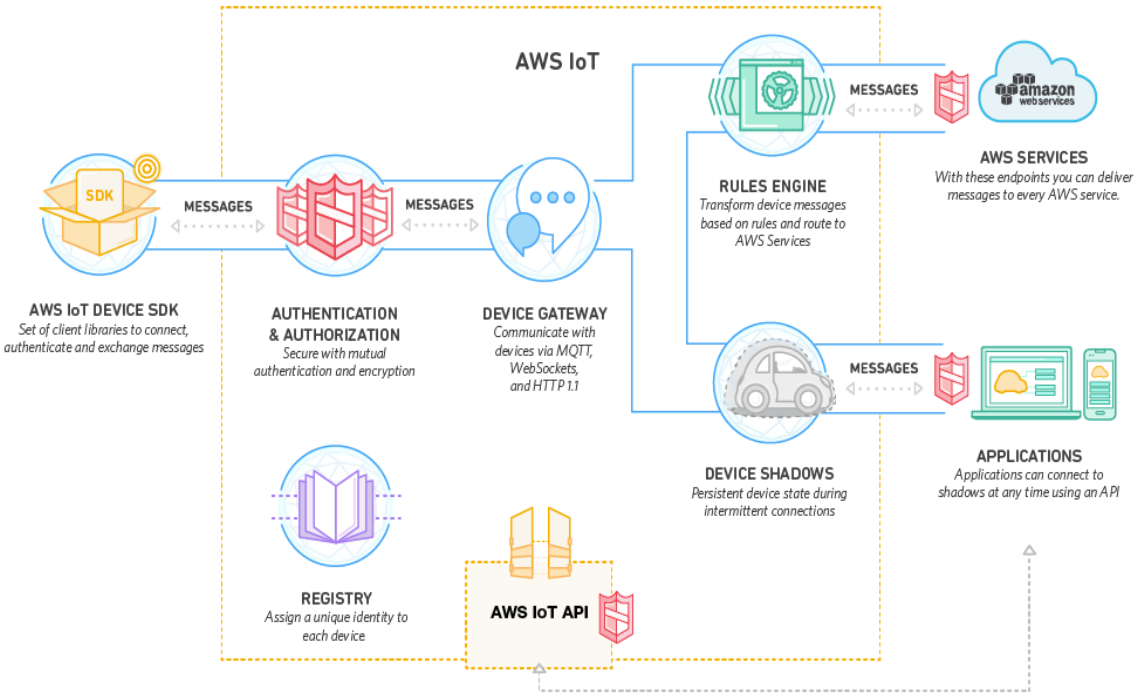


Рисунок 3.3 – – Схема роботи AWS IoT [12]

AWS також нещодавно оголосила про Kinesis Analytics, яку можна використовувати для аналізу потоків за допомогою SQL-подібної мови.

Кожен пристрій, який з'єднується з AWS, представлений як Shadow пристрій. Shadow підтримує ідентифікацію та останній відомий стан певного пристрою та забезпечує канал для надсилання й отримання повідомлень. Коли повідомлення надсилається на пристрій, AWS гарантує доставку повідомлення, якщо пристрій знаходиться в автономному режимі, воно буде доставлене, коли пристрій знову з'єднається.

3.4 Хмарна платформа ThingSpeak

ThingSpeak - це відкрита платформа IoT даних, в основі якої лежать публічні хмарні технології.

ThingSpeak - проект з відкритим вихідним кодом. Це платформа і API для зберігання та вилучення даних від пристроїв по HTTP протоколу через інтернет або локальну мережу. З ThingSpeak можна створювати додатки контролю даних від різних датчиків, додатки що відстежують місце

розташування, можна навіть побудувати "соціальну мережу речей" з оновленнями статусів.

У ThingSpeak можна інтегрувати популярні пристрої і сервіси такі як:

- ESP8266, Arduino та Raspberry Pi
- Мобільні і Web додатки
- Соціальні мережі
- Аналіз даних в MATLAB

Основу платформи складають канали, в які і надсилаються дані для зберігання і візуалізації. Кожен канал включає в себе 8 полів для будь-якого типу даних, 3 поля для розташування (широта, довгота, висота), і 1 поле стану. Як тільки буде зареєстровано в ThingSpeak канал відразу можливо відправляти туди дані, обробляти їх і отримувати до них доступ користувацькими додатками. Канали підтримують формати даних JSON, XML і CSV. Дані відправляються в ThingSpeak HTTP POST запитом.

ThingSpeak співпрацює з MathWorks, Inc. MathWorks Корпорація та інтегрована підтримка програмного забезпечення MATLAB орієнтованого на чисельні обчислення. Близькі стосунки з MathWorks дозволяють користувачам використовувати можливості MATLAB для візуалізації та аналізу завантажених даних без необхідності придбання ліцензії MATLAB. Веб-сайт документації ThingSpeak є частиною веб-сайту MathWorks, тому облікові записи користувачів MathWorks можуть використовуватись для входу на сайт ThingSpeak. Також «ThingSpeak Панель інструментів підтримки» доступна для настільних MATLAB для аналізу та візуалізації даних, що зберігаються на платформі ThingSpeak (будь-яка веб-версія або приватна установка).

Візуалізація та аналіз

Для перетворення та візуалізації даних або активації дій в ThingSpeak, використовуються так звані додатки. Візуалізація завантажених даних

можлива за допомогою простого вбудованого настроюваного лінійного графіка або з додатком для візуалізації MATLAB. Ця програма дає можливість створити інтерактивну або статичну візуалізацію з використанням функцій MATLAB. MATLAB код можна створити з одного з підготовлених шаблонів і редагувати в редакторі вбудованого коду з виводом зображення безпосередньо під ним. Інший спосіб показати кастомні візуалізація за допомогою додатку плагінів. Для цього плагіни дозволяють користувачеві використовувати HTML, CSS та JavaScript. Два шаблони JavaScript вже готові до використання – Google датчик і діаграма з кількома серіями. Код MATLAB можна також запустити в додатку MATLAB Analysis, який призначений для різних розрахунків, перетворень даних тощо. На рисунку зображено графік температури та вологості за декілька днів.

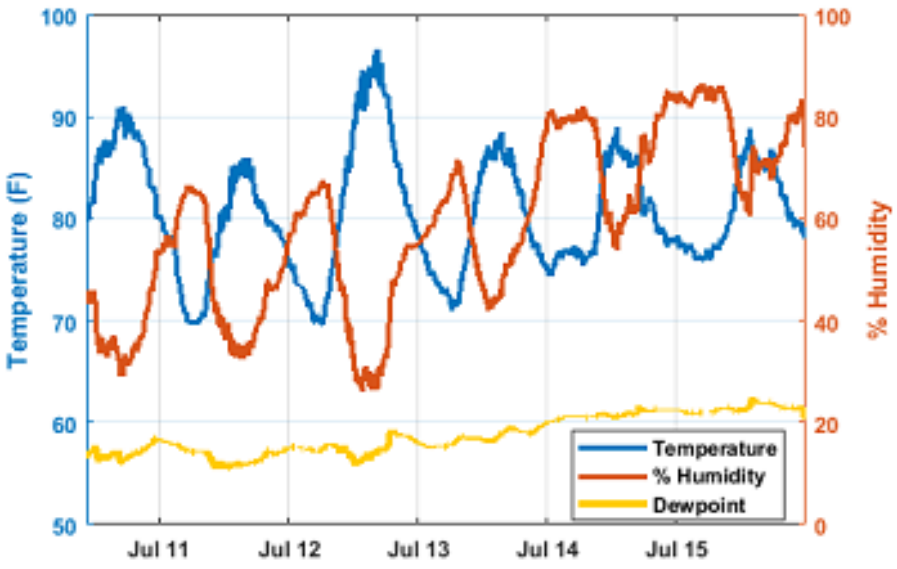


Рисунок 3.4 – Приклад візуалізації в ThingSpeak[13]

Канали поділяються на дві категорії: загальні та приватні. За замовчуванням, канал є приватним і вимагає наявності ключа Read API для доступу до його потоку. Власник каналу може зробити загальнодоступний канал, який дає можливість іншим користувачам використовувати його канали без Read API ключа. Ключы API для запису та читання дозволяють записувати та читати дані до або з (приватного) каналу за допомогою ThingSpeak API або MATLAB коду. Ключ API - 16-значний автоматично

згенерований рядок, ключі Read та Write API можуть бути регенеровані в налаштування каналу, з можливістю наявності декількох ключів Read API для різних додатків. Ключі API повинні бути вказані в кожному HTTP запиті або функції MATLAB. Ключ API користувача використовується для викликів API, наприклад, створення нового каналу,.

Веб-застосунок ThingSpeak та виклики API захищені протоколом HTTPS.

3.5 Інші хмарні платформи

Існує велика кількість хмарних платформ для керування мережею IoT пристроїв, збереження, обробки та візуалізації даних. Інформація про деякі з них наведена в таблиці 3.1

Таблиця 3.1 – Не висвітлені хмарні платформи

Платформа	Розробка додатків	Управління пристроями	Аналітика	Візуалізація
Carriots	-	+	-	+
IoT від IBM	+	+	-	+
ThingWorx	+	-	-	+
Oracle IoT	+	+	+	+
Eclipse IoT	+	+	-	-
Thethings.io	+	-	+	+
OpenHAB	+	-	+	-

3.6 Висновки

Отже, було розглянуто архітектуру хмарних платформ, оскільки це доволі нове поняття можна зробити висновок, що немає чіткої термінологіїю Але все ж можна виділити такі поняття як: керування пристроями, збереження даних, обробка та візуалізація, підтримка розробки додатків.

Надалі, проаналізовано такі платформи як Azure IoT, Google Cloud Platform, AWS IoT, ThingSpeak. Платформа Azure також набирає популярність за рахунок широкого функціоналу та можливості розширення. AWS надає гарні можливості для обробки даних, за рахунок існуючих додатків та легкої SQL подібної мови. Google Cloud Platform дає змогу користувачам використовувати клієнтьскі бібліотеки, що існують для багатьох мов програмування. Так само як і Microsoft з Windows IoT, Google створив свою операційну систему для IoT. З точки зорку можливостей для розширення платформа ThingSpeak має перевагу, так як існує багато засобів для аналізу та моніторингу даних.

4 КЕРУВАННЯ РОЗУМНИМИ РЕЧАМИ

4.1 Керування пристроями IoT

Існує чотири основні вимоги щодо керування пристроєм IoT: забезпечення та перевірка автентичності, конфігурація та контроль, моніторинг та діагностика, а також оновлення та технічне обслуговування програмного забезпечення. [14]

Після встановлення пристрою Інтернету речей використання сценарію "fire and forget" недопустиме. Потрібні виправлення помилок та оновлення програмного забезпечення; деякі пристрої не зможуть відремонтувати або замінити.

Будь-яка система IoT має вирішувати чотири основні категорії керування пристроями:

- Забезпечення та перевірка автентичності
- Конфігурація та контроль
- Моніторинг та діагностика
- Оновлення програмного забезпечення та технічне обслуговування

За даними IDC, кількість підключених до Інтернету пристроїв, як очікується, до 2020 року досягне 30 млрд. [15] Багато виробників пристроїв значною мірою недосвідчені в питаннях безпеки. Нещодавно лідер хакерів РН Джойс заявив, що IoT це чудовий спосіб атакувати ціль, оскільки пристрої забезпечують спосіб введення в мережу, який часто блокуються системними адмінстраторами. [18]

Аутентифікація пристрою - це акт надійного визначення ідентичності пристрою, який гарантує довіру до нього. Хмарна служби, до якої підключаються пристрої, повинна знати, що пристрій насправді є аутентичним пристроєм, працює надійний програмний продукт і працює від імені надійного користувача.

Забезпечення - це процес занесення пристрою в систему. Автентифікація є частиною цього процесу, де зареєстровано лише пристрої, які представляють належні облікові дані. Точна інформація про цей процес може сильно відрізнятися залежно від реалізації. Проте в більшості додатків пристрій, що розгортається, завантажується сертифікатом або ключ(зберігається в захищеній області пам'яті), який ідентифікує його як справжній, також пристрій знає URL-адресу сервера для підключення, щоб зареєструватися самостійно. Коли пристрій спочатку підключений до локальної мережі, він "дзвонить додому", а потім, на основі, такої інформації як модель та серійний номер пристрою, він може отримати додаткові дані конфігурації.

У більшості випадків пристрій буде додатково налаштований кінцевим користувачем з такими атрибутами, як ім'я та місце розташування та налаштування програми. У прикладі керування флотом, пристрій використовується для відстеження місцезнаходження та певної телеметрії на транспортному засобі та передачу цієї інформації назад у хмару через стільникове з'єднання. Деякі параметри повинні бути записані після встановлення пристрою, наприклад унікальний ідентифікатор причепа або вантажного автомобіля (можливо, номерний знак або VIN номер). Інші налаштування конфігурації, такі період часу між повідомленнями про стан завантаження, також визначаються та запрограмовані в пристрій.

Щоб застосувати певну керуючу здатність до системи, необхідно вміти дистанційно скинути налаштування пристрою, щоб досягти добре відомих станів і відновити помилки та впровадити нові зміни в конфігурації. Можливо, також скинути налаштування пристрою за замовчуванням, це корисно, коли потрібно відключити пристрій або необхідно відновитись після невідомих помилок. Нарешті, дуже важливо видавати команду для оновлення або перезавантаження вбудованого програмного забезпечення, щоб забезпечити безпеку віддаленого пристрою, впроваджувати функції та виправляти помилки.

Моніторинг та діагностика

У системі тисяч віддалених пристроїв, плавна і безпечна робота кожного пристрою може безпосередньо впливати на фінансову складову компанії. Моніторинг та діагностика є життєво важливими для мінімізації впливу будь-якого простою пристрою через програмні помилки або інші непередбачені операційні проблеми.

Можливість завантажувати журнальні дані також важлива для діагностики та вирішення програмних помилок. Розробник додатків повинен реалізувати реєстрацію помилок в журналі виведення, а програмне забезпечення для керування пристроєм має зробити цю інформацію доступною для завантаження, коли трапляється помилка.

Обслуговування та оновлення програмного забезпечення

Для більшості розробників продукту, особливо компаній, які намагаються швидко вийти на ринок даний аспект може ігноруватись. Однак це один з найважливіших аспектів управління пристроєм - абсолютно необхідно безпечно оновлювати та підтримувати віддалене програмне забезпечення пристрою.[14]

4.2 Керування CAN пристроями

Протокол CAN описує тільки передачу даних від одного вузла до іншого. Іншими словами, він описує тільки два нижніх рівня еталонної моделі OSI. Немає ніякої інформацій як обробляти поле даних, яким чином використовувати поле ідентифкатора, та що робити з даними розмір яких більший, ніж 8 байт. Механізми роботи які не описано стандартом CAN зазвичай називають протоколоми виского рівня CAN HLP(High Level Protocol)

Зазвичай вони описують наступні пололження:

- Процедура пошуку нового вузла
- Засоби використання ідентифікатора вузла
- Формат повідомлень

- Яким чином обробляти помилки на прикладному рівні
- Механізм відправки даних (періодичні, по запиту, негайні повідомлення

4.2.1 Керування CAN пристроями з CAnKingdom

CAnKingdom розроблений для керування системами пристроїв. Це набір примітивів, що використовуються поверх CAN протоколу, а також засіб для інженерів для створення оптимізованої системи. Призначений для застосування в стаціонарних або переносимих системах, що потребують контролю в реальному часі. CanKingdom розроблено в концепції, що характеризується як зрозуміла, безпечна, проста та ефективна.

В даному протоколі мережа представляю собою королівство, в якому столиця і міста – це аналоги різних пристрої та вузли. У кожного міста є мер, який відподвідає за нього, а також CAN мережа, представлена поштою. На рисунку 4.1 зображена традиційна модель CAnKingdom

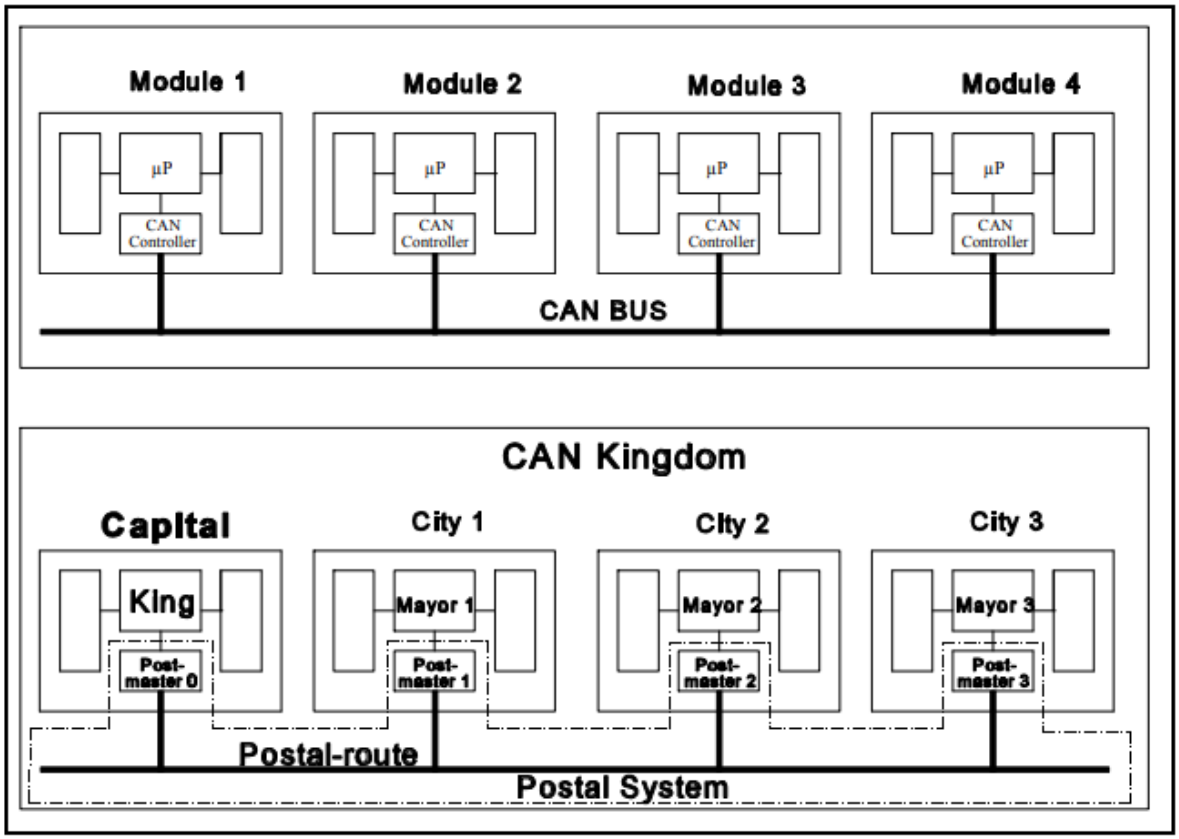


Рисунок 4.1 – Модель CAnKingdom[18]

CanKingdom спроектований для побудови стабільних та надійних систем. Завдяки розділенню на системи та модулі, проектувальник може повністю контролювати та нести відповідальність за кінцеву систему. Також система може бути переналаштована завдяки керуючим повідомленням. Таким чином можливо модернізувати систему після встановлення.

4.2.2 Керування CAN пристроями з CAL and CANopen

CANopen – це протокол високого рівня, створений для використання у вбудованих системах, наприклад автомобліях. Специфікація покриває собою засоби для мережі, опису пристроїв, визначення інтерфейсів. CANopen надає стандартизований спосіб для комунікації між пристроями та додатками від різних виробників.

Одна з головних концепцій протоколу є словиники об’єктів, що представляють собою таблиці, які зберігають конфігурації та дані. Стандарт визначає 16-бітні індекси та 8-бітні підіндекси. Таким чином, можливо мати до 65536 індексів і 256 підіндексів на кожний індекс. Стандарт визначає, що деякі адреси або діапазони адрес повинні містити конкретну інформацію. Наприклад, відповідно до стандарту індекс 1008, підіндекс 0, повинен містити ім’я пристрою. Будь-який CANopen пристрій може вичитати цей індекс з мережі і отримати дану інформацію. Мінімальний набір індексів специфікації визначається для пристрою,щоб можна було вважати його CANopen сумісним.

Основні типи даних, що входять до словника об'єкта: булеві, цілі беззнакові числа, цілі знакові числа, плаваюча точка та символ. Більші складні типи даних, такі як рядки, дата та час, можуть бути побудовані з основних типів даних. Ці типи даних можуть бути використані для визначення спеціальних типів даних, специфічних для CANopen, таких як Запис параметрів PDO / SDO та параметри відображення PDO.

Форамат повідомлення базується на стандартному CAN. Де CAN-ID розділяється на 4 біти функціонального коду та 7 біт ідентифікатора вузла.

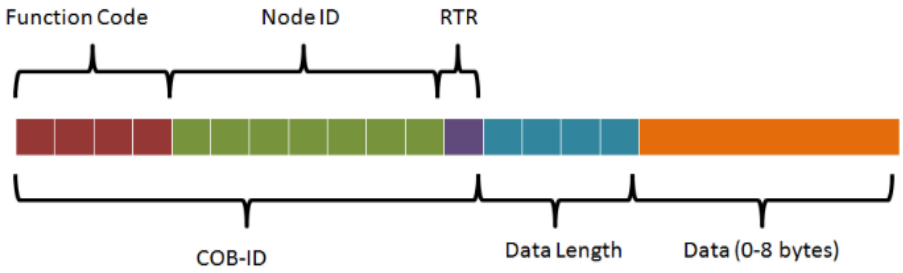


Рисунок 4.2 – Формат CANopen повідомлення[18]

Всі COB-ID повинні бути унікальними, щоб ункикнути конфліктів у шині.

Специфікація зазначає, що кожний вузол мережі повинен визначити обробляти операції читання та запису свого словника об’єктів. Механізм прямого доступу до словника забезпечуються сервісним об’єктом даних – Service Data Object. Вузол, що опитує називають SDO клієнтом, а інший – SDO сервером.

Як правило, основний вузол CANopen надсилає запит до мережі, а відповідний вузол відповідатиме на запитувані дані. CANopen використовує зарезервовані ідентифікатори повідомлень, щоб полегшити комунікацію. Коли клієнт SDO хоче запитати інформацію з сервера, він надсилає запит SDO, використовуючи CAN-ID 600h + ідентифікатора вузла. Після цього сервер відповість за допомогою CAN-ID 580 x + ідентифікатора вузла. Ідентифікатор вузла визначає, який підпорядкований вузол отримає повідомлення. Для прикладу основний вузол (клієнт SDO) надсилає повідомлення до мережі з CAN-ID 603h. Хоча всі вузли бачать це повідомлення, всі вузли, окрім цільового вузла, ігнорують його, тому що повідомлення не призначене для них. Цільовий вузол розуміє, що повідомлення з ідентифікатором 603h означає, що повідомлення призначене для цього вузла і є запитом SDO. У полі даних повідомлення буде вказано індекс та суб-індекс об'єкта, до якого головний вузол хоче отримати доступ. Цільовий вузол потім відповідає

ідентифікатору повідомленням 583h. Поле даних відповідного повідомлення буде містити потрібні дані. Структура повідомлення зображена на рис. 4.3.

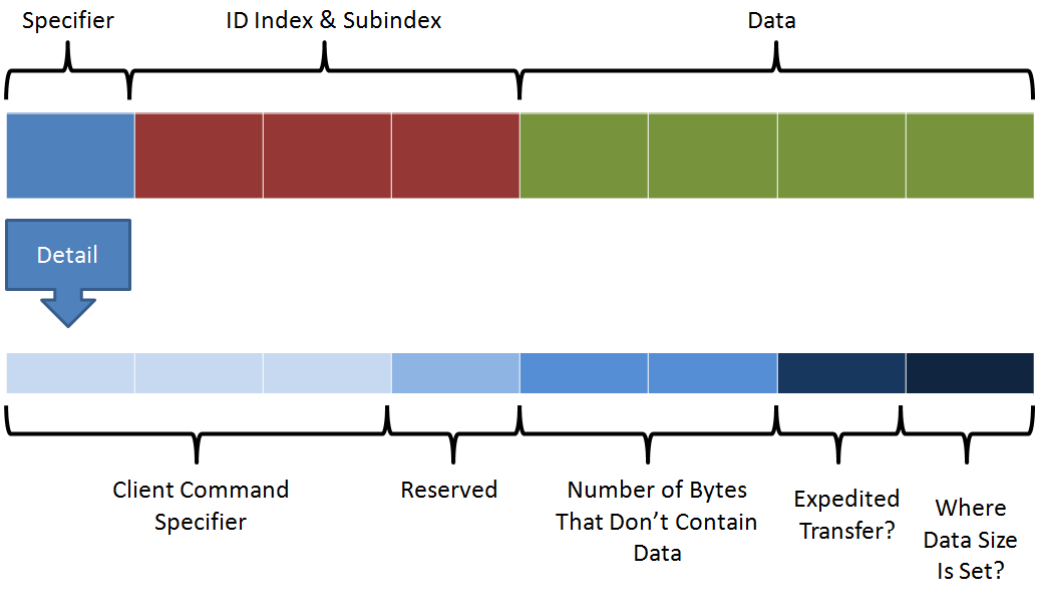


Рисунок 4.3 – SDO формат повідомлення[18]

Секція даних повідомлення містить специфікатор, індекс та підіндекс словника об'єктів, поле розміру даних та самі дані.

Обробка даних представляє собою інформацію, яка може змінюватись з часом. Для того, щоб спростити механізм – запит-відповідь як в SDO повідомленнях, надається можливість відправлять об'єкти обробки даних Process Data Objects (PDO). Існує два типи PDO: отримання та передачі. Тобто ті дані, що вузол буде надсилати та отримувати. В доповнення, існують ще два параметри: конфігурація та відображення. Існують різні способи ініціювати передачу PDO, наприклад через подію, деякий час, спосіб вказується в секції конфігурації.

Сервіс керування мережею включає в себе можливість змінювати стан вузла між такими, як ініціалізація попередньо-експлуатаційний, експлуатаційний та зупинений. NMT дає змогу контролювати індивідуальні вузли, наприклад попередньо-експлуатаційний стан зазвичай використовується для налоголження пристроїв, тому PDO повідомлення не дозволяються у цьому стані. Спкцифікація CANopen вимагає від пристрою

деякого способу, що зазначає чи є він «живим». Для цього використовуються повідомлення типу heartbeats. Якщо дане повідомлення не прийшло за деякий час, головний вузол може виконати необхідні дії. Повідомлення визначається за допомогою CAN-ID 0x700 + ідентифікатор вузла. А переше поле данних рівне 0x60.

4.2.3 Керування CAN пристроями з SDS

SDS – протокол високого рівня, що визначає для користувачів операції читання, запису, дії та події. Читання та запис працюють над атрибутами деякого пристрою. Дії повідомляють пристрою про необхідність виконання деякої операції. Події дають змогу повідомляти асинхроно деяку інформацію. Перший байт повідомлення зазвичай містить специфічну для протоколу інформацію, наприклад чи є повідомлення запитом, чи відповіддю, або відповіддю з помилкою, чи є повідомлення частиною фрагменту або більшого повідомлення, а також який із 16 об’єктів використовуються. Наступний байт зазначає номер атрибута, який вичитуюється або записується, або номер дії чи події. Інші байти використовують для передачі даних.

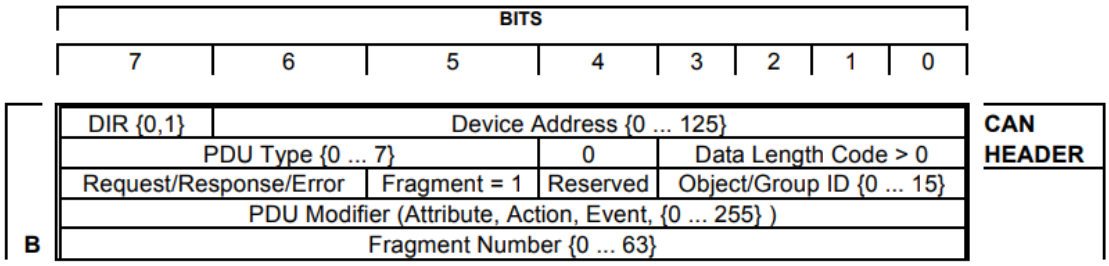


Рисунок 4.4 – SDS повідомлення[18]

Протокол зазвичай використовують в об’єктно-орієнтованій методології. Включаючи ієрархію пристроїв з наслідуванням. Усі SDS пристрої наслідуються від одного загального об’єкту пристроя, який містить мінімально необхідний набір функцій для того, щоб пристрій міг спілкуватись

в мережі SDS. Цей підхід дає можливість різним виробникам систем робити пристрої, що базуються на спільному об’єкті.

Основними перевагами для замовника стали спрощення встановлення та зниження загальних витрат, в той же час забезпечується гнучка та зручна конфігурація та обслуговування системи.

4.3 Висновки

Розглянуто основні принципи керування IoT мережами, серед них можна визначити такі: забезпечення та перевірка автентичності, конфігурація та контроль, моніторинг та діагностика, а також оновлення та технічне обслуговування програмного забезпечення.

Оскільки CAN мережа працює тільки на фізичному і канальному рівні моделі OSI, для керування мережею «розумних» речей необхідне використання протоколів вищого рівня. Проаналізовано протоколи CANKingdom, CANOpen, SDS. Серед них CANOpen має найбільше переваг, так як існує вже багато доступних пристроїв, що підтримують цей стандарт, також специфікація гарно задокументована і наявні приклади реалізації. Таким чином доцільно використовувати CANOpen для побудови тестової моделі.

5 РОЗРОБКА МОДЕЛІ КЕРУВАННЯ РОЗУМНИМИ РЕЧАМИ

Модель представляє собою мережу «розумних» пристроїв. Головний пристрій виступає в ролі шлюзу, іншим є датчик. Як фізичний протокол для обміну даними між датчиками та головним пристроєм обрано протокол CAN. Головний пристрій під'єднується до мережі інтернет за допомогою Ethernet або WiFi. Абстрактно модель зображена на рисунку 5.1.

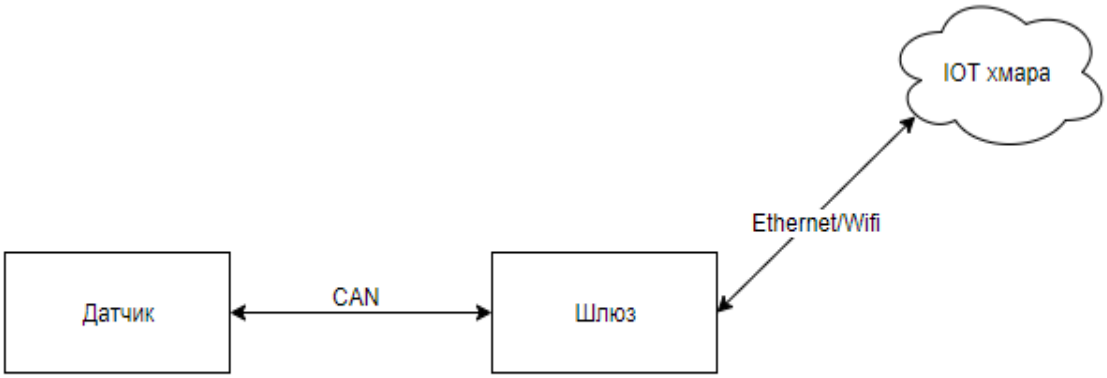


Рисунок 5.1 – Модель мережі розумних речей

Отже, для демонстрації концепції, побдуємо систему, яка буде вимірювати вологість та температуру навколишнього середовища.

5.1 Апаратна платформа

Для реалізаці системи обрані наступні пристрої:

- Шлюз – Raspberry PI 3 B+
- Датчик побудовано на Arduino Uno

Raspberry Pi 3 B+ побудований на системі-на-кристалі, що має вбудовану підтримку WiFi та Gigabit Ethernet. Недоліком є відсутність підтримки протоколу CAN. Для того, щоб додати цю можливість було використано SPI CAN модуль MCP2515 та TJA1050. Оскільки даний модуль має логічний

рівень 1 при 5 В, а Raspberry використовує 3.3 В для логічної одиниці було також додано логічний перетворювач рівнів з 3.3 В до 5В і навпаки.

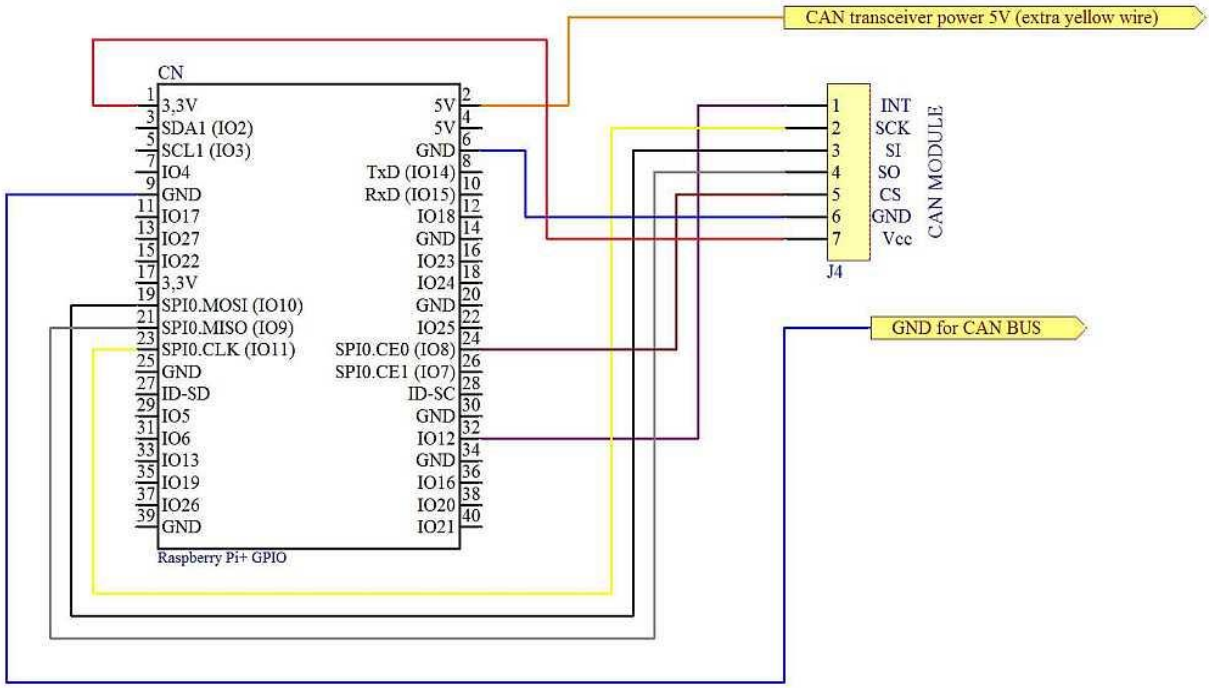


Рисунок 5.2 – Схема підключення CAN трансивера до Raspberry

Датчик побудовано на широкорозповсюдженій апаратній платформі Arduino Uno. Для виміру температури і вологості використовується датчик GY-21 HTU21, який під’єднано до Arduino по I2C. Arduino також не має вбудованої підтримки мережі CAN, тож під’єднано SPI CAN модуль MCP2515 та TJA1050. Логічний перетворювач рівнів не потрібний, адже Arduino використовує 5В для логічного рівня 1.

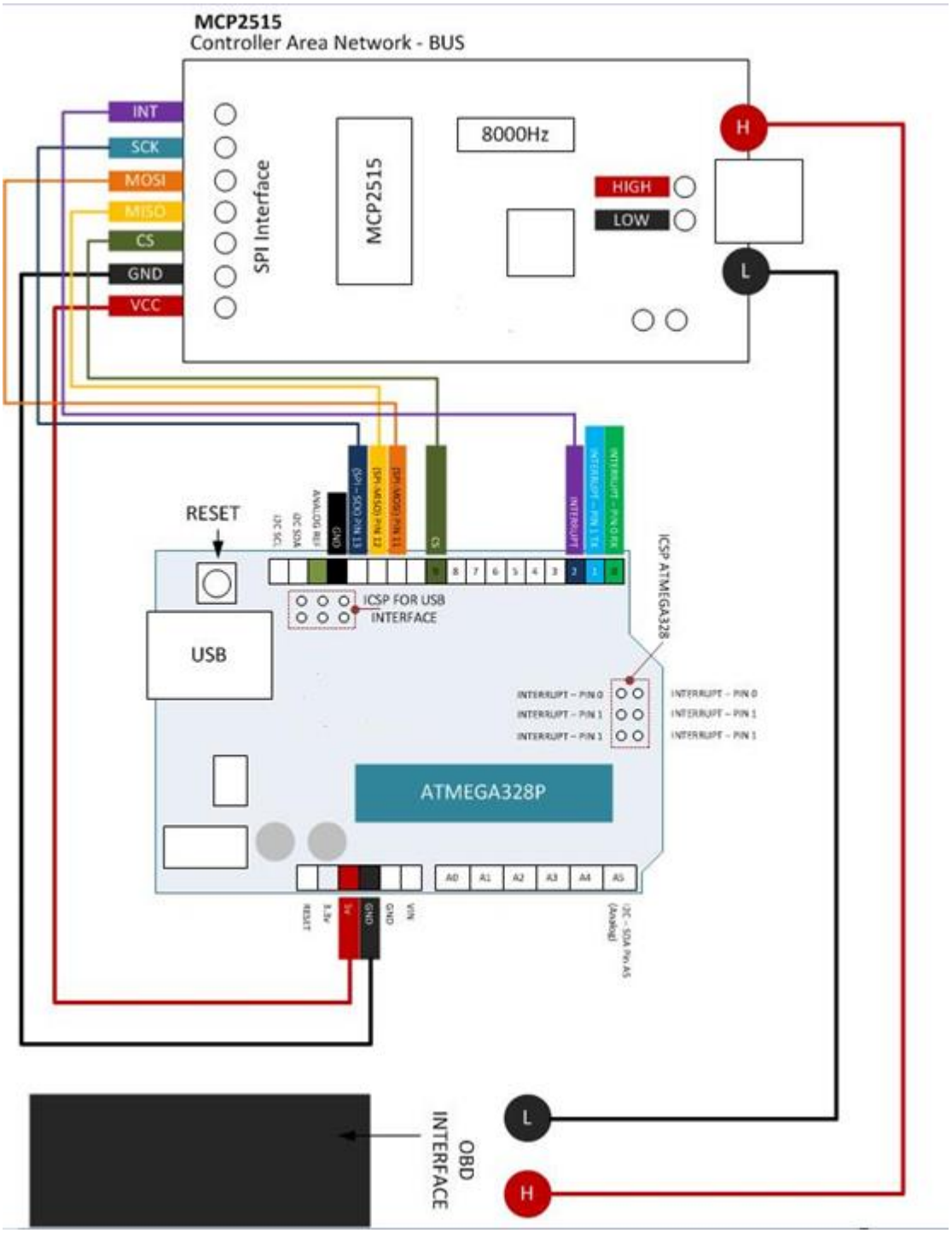


Рисунок 5.3 – Схема підключення датчика

5.2 Програмна платформа

Для взамодії пристроїв використано наступні додатки:

- ThingSpeak розгорнуто на сервері
- CanOpenNode для керування CAN пристроями з Raspberry
- Програма з підтримкою протоколу CanOpen для Arduino

- Додаток узгодження керування CanOpen та Rest ThingSpeak

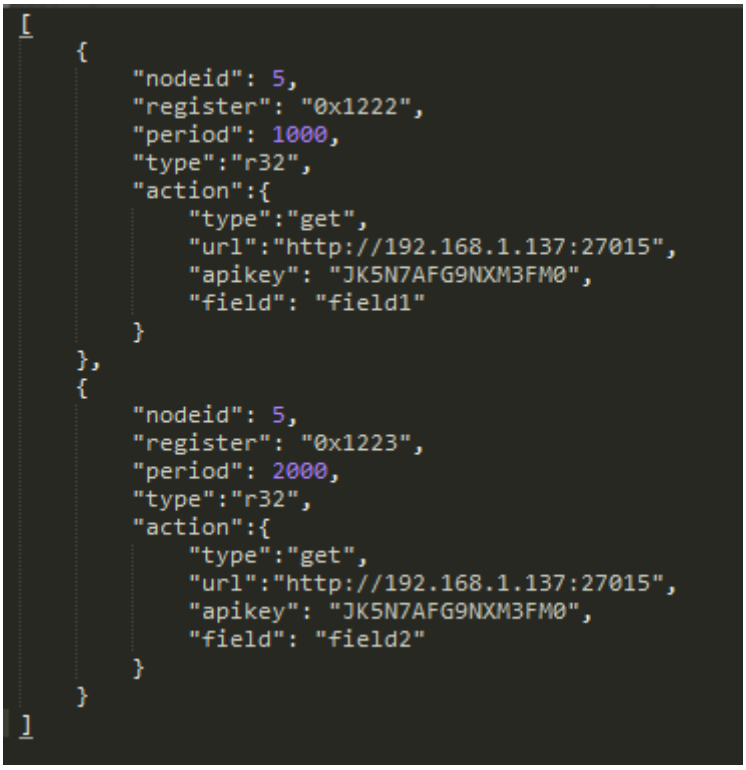
Для керування CAN пристроями необхідно використовувати протоколи високого рівня, наведені в попередньому розділі. Таким чином було обрано відкриту реалізацію протколу CanOpen – CanOpenNode[20]. Додаток надає можливість вичитувати та записувати дані з під’єднаних CAN пристроїв, що підтримують протокол CanOpen.

Програмна прошивка Arduino не має вбудовнаї підтримки протоколу CanOpen. Отже, необхідно розробити додаток для Arduino, який надав би змогу вичитувати дані з датчика температури та вологості та віддавати їх через вищезазначений протокол.

Додаток узгодження керування CanOpen та Rest ThingSpeak необхідний для передачі отриманих даних з датчика через протокол CanOpen на хмарний ThingSpeak, через Rest.

5.3 Результати роботи

Розглянемо конфігураційний файл для узгодження передачі даних з CAN датчика зображений на рисунку 5.4. Даний датчик описаний відповідно до специфікації протоколу CanOpen. Ідентифікатор вузла рівний 5. Поле «register» визначає положення даних в Object Dictionary пристрою. Поле «type» визначає тип даних, що зберігається в даному регістрі, в даному випадку «r32», що віповідає типу числа з плаваючою точкою стандарту IEEE754. Поле «action» визначає, яким чином передати дані на сервер ThingSpeak.



```
[
    {
        "nodeid": 5,
        "register": "0x1222",
        "period": 1000,
        "type":"r32",
        "action":{
            "type":"get",
            "url":"http://192.168.1.137:27015",
            "apikey": "JK5N7AFG9NXM3FM0",
            "field": "field1"
        }
    },
    {
        "nodeid": 5,
        "register": "0x1223",
        "period": 2000,
        "type":"r32",
        "action":{
            "type":"get",
            "url":"http://192.168.1.137:27015",
            "apikey": "JK5N7AFG9NXM3FM0",
            "field": "field2"
        }
    }
]
```

Рисунок 5.4 – Конфігураційний файл для програми узгодження керування «розумними» речами.

Після запуску датчика та шлюза, та необхідних налаштувань, запустимо програму узгодження керування «розумними» речами.

```
pi@raspberrypi:~/diploma $ python layer.py
/home/pi/CANopenSocket/canopencomm/canopencomm 5 read 0x1222 0 r32
[1] 24.8473

http://192.168.1.137:27015/update?key=JK5N7AFG9NXM3FM0&field1=24.8473

/home/pi/CANopenSocket/canopencomm/canopencomm 5 read 0x1223 0 r32
[1] 41.4625

http://192.168.1.137:27015/update?key=JK5N7AFG9NXM3FM0&field2=41.4625

/home/pi/CANopenSocket/canopencomm/canopencomm 5 read 0x1222 0 r32
[1] 24.8473

http://192.168.1.137:27015/update?key=JK5N7AFG9NXM3FM0&field1=24.8473

/home/pi/CANopenSocket/canopencomm/canopencomm 5 read 0x1222 0 r32
[1] 24.8687

http://192.168.1.137:27015/update?key=JK5N7AFG9NXM3FM0&field1=24.8687

/home/pi/CANopenSocket/canopencomm/canopencomm 5 read 0x1223 0 r32
[1] 41.4701

http://192.168.1.137:27015/update?key=JK5N7AFG9NXM3FM0&field2=41.4701
```

Рисунок 5.5 - Результати роботи програми узгодження протоколів керування

В той же час розглянемо активність на CAN шині зображену на рис. 5.6

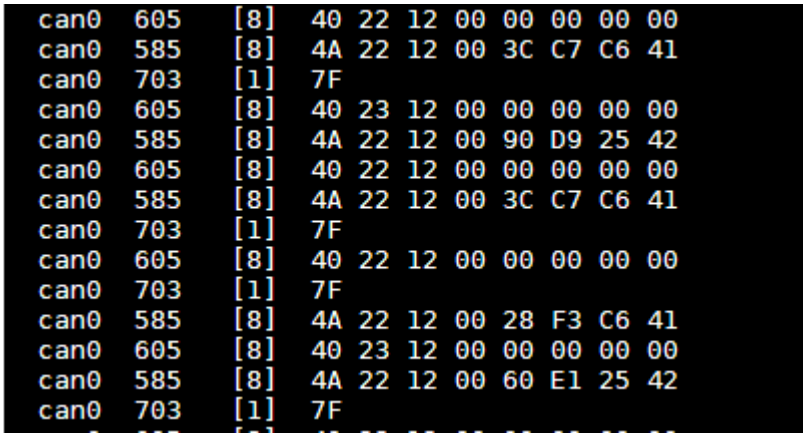


Рисунок 5.6 - Повідомлення в CAN шині

Таким чином з запуском додатку можна спостерігати зростаючу активність на шині CAN. Відповідно до специфікації CanOpen повідомлення з ідентифікатором «605» - це запит на зчитування значення вузла з порядковим номером 5. На нього датчик відповідає повідомленням з ідентифікатором «585».

Під час роботи програми, зчитані з датчиків дані поступають на платформу ThingSpeak. На рис. 5.7 зображено відповідне оновлення графіків температури та вологості.

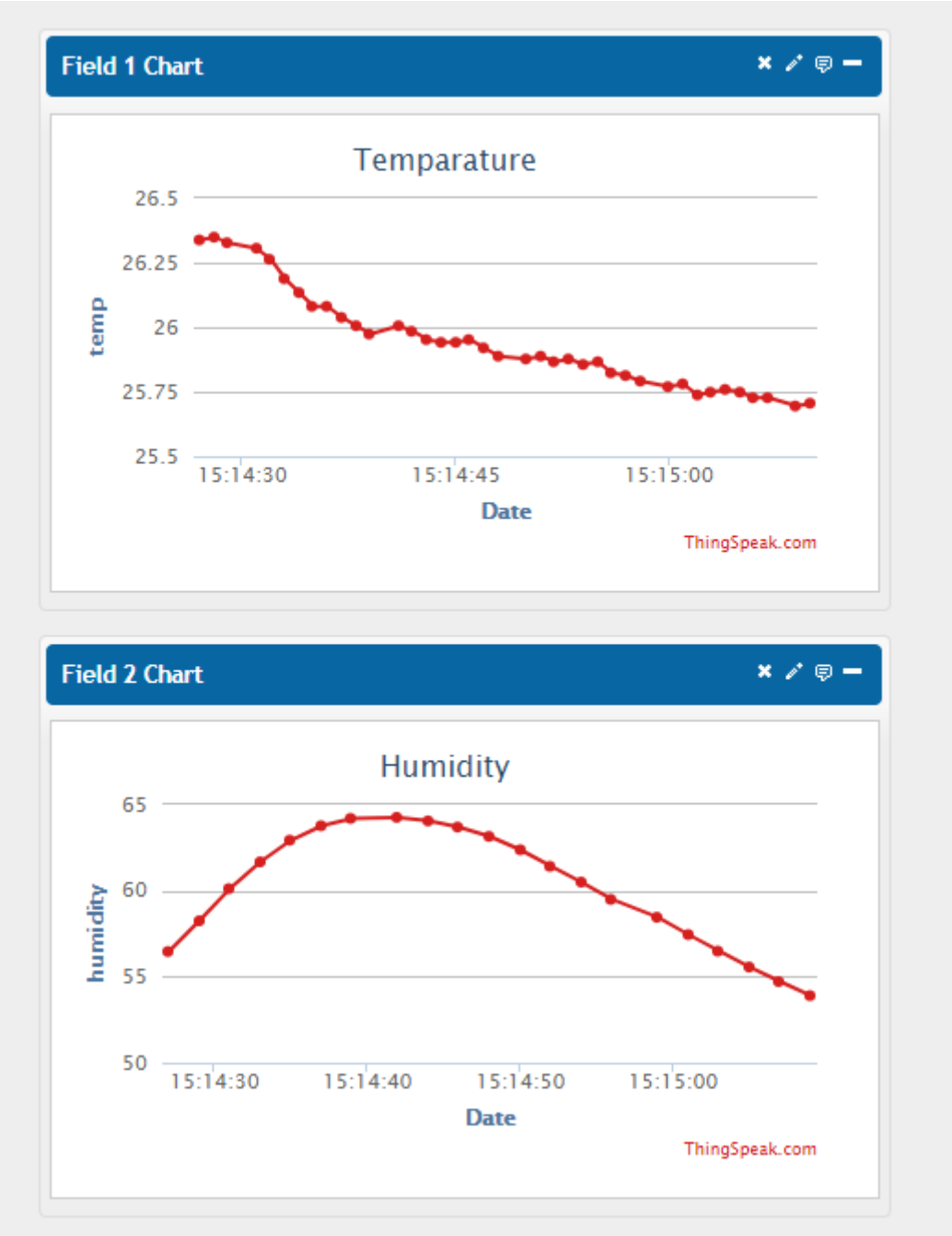


Рисунок 5.7 - Графік температури та вологості на веб сторінці ThingSpeak

5.4 Висновки

Отже, було обрано необхідні апаратні пристрої для побудови моделі CAN мережі розумних речей. Наведено схеми підключення та налаштування пристроїв.

Також наданий перелік необхідного програмного забезпечення для роботи моделі. Створено додаток для Arduino, який підтримує зчитування з

підключеного дадчика та передачу необхідної інформації через протокол високого рівня CanOpen в мережі CAN. Розгорнуто локально хмарну платформу ThingSpeak і проведено необхідні для моделі налаштування.

Розроблено програму узгодження протоколів CanOpen та HTTP Rest. Для роботи програми описано конфігураційний файл, що надає можливість вичитувати коректні дані з датчика та передавати їх в відповідний канал платформи ThingSpeak.

Таким чином, модель відповідає поставленому завданню і демонструє успішність виконання розроблених додатків.

6. РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

Метою даного розділу є проведення маркетингового аналізу стартап проекту для визначення принципової можливості його ринкового впровадження та можливих напрямів реалізації цього впровадження.

6.1 Опис ідеї проекту

У даному розділі описано економічне обґрунтування реалізації стартап-проекту на тему “ Узгодження протоколів керування мережею "розумних" речей для хмарного Thing Speak та CAN internet of things.”.

Основними перевагами системи "розумних" пристроїв є можливість автоматизації їх роботи та віддаленого контролю. Це дозволяє автоматично контролювати параметри мікроклімату у приміщенні, вмикати та вимикати різного роду пристрої, надсилати повідомлення користувачам при зміні параметрів та навіть автоматично розміщувати пости у соціальних мережах.

Таблиця 6.1 – Опис ідеї стартап-проекту

Зміст ідеї	**Напрямки застосування**	**Вигоди для користувача**
Розробка IoT системи, яка дозволяє керувати пристроями за допомогою протоколу CAN та надсилати дані на IoT платформу	1. Система розумного дому	Дозволяє контролювати параметри у приміщенні, встановлювати бажані значення, переглядати статистику
	2. Промисловість	Надійний засіб управління пристроями, аналізу даних

Засоби автоматизації та управління розумними «речами» стають здебільш популярними, як серед звичайних користувачів так і у промсловому застосуванні. Даний стартап-проект, пропонує користувачам надійний засіб

автоматизації та управління «розумними пристроями» об’єднаних за допомогою протоколу CAN.

У таблиці 6.2 наведено сильні, слабкі та нейтральні характеристики ідеї проекту.

Таблиця 6.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	**Технікоекономічні характеристики ідеї**	**(потенційні) товари/концепції конкурентів**				**W (слабка сторона)**	**N (нейтральна сторона)**	**S (сильна сторона)**
		Мій проект	**Конкурент1**	**Конкурент2**	**Конкурент3**			
1	Форма використання	Веб	Веб	Мобільний клієнт	Веб + мобільний телефон	+		
2	Собівартість	Низька	Низька	Низька	Висока			+
3	Наявність інтернету	+	+	+	+		+	
4	Надійний протокол для об’єднання пристроїв	+	+	-	-			+
	Підтрмка інших протоколів фізичного рівня (BLE, 6LOWPAN, ZigBEE)	-	-	+	+	+		

Сильною стороною проекту є можливість під’єднання розумних пристроїв в мережу CAN. Слабкою стороною підримка інших протоколів фізичного рівня, відсутність мобільного додатку. Всі інші характеристики є нейтральними. Тому даний проект можна вважати конкурентоспроможним.

6.2 Технологічний аудит ідеї проекту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару).

Таблиця 6.3 – Технологічна здійсненність ідеї проекту

№ п/п	**Ідея проекту**	**Технології її реалізації**	**Наявність технологій**	**Доступність технологій**
1	Створення вебдодатку	Flask	Наявна	Безкоштовна, доступна
		SocketCan	Наявна	Безкоштовна, доступна
2	Протокол взаємодії	MQTT	Наявна	Безкоштовна, доступна
		REST	Наявна	Безкоштовна, доступна
3	IoT платформа	ThingSpeak	Наявна	Безкоштовна, доступна
4	Апаратна платформа	Raspberry PI	Наявна	Платна, доступна
		Arduino	Наявна	Платна, доступна
		Трансивер CAN2515	Наявна	Платна, доступна

Обрана технологія реалізації ідеї проекту: Для апаратної реалізації - Raspberry PI + CAN2515, так як дані пристрої широко розповсюджені та розробка на них доволі проста. Для веб додатку та протоколу взаємодії буде використано Flask та Rest відповідно.

6.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення основних можливостей, які відкриваються при запуску проекту, а також, які заходи потрібно провести для запобігання загроз. Розглянуто у якому напрямку розвивається ринок, а також потенційні цільові групи клієнтів. Попередню характеристику потенційного ринку стартап-проекту наведено у таблиці 6.4.

Таблиця 6.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	5000 грн./ум.од
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	R = (3000000 * 100) / (1000000 * 12) = 25%

Розглянуто основні характеристики. Існують всі умови для виходу на потенційний ринок. Невелика кількість конкурентів може гарантувати перспективний вихід та розвиток.

Надалі, визначається цільова група клієнтів. Таким чином можливо сфокусуватись на важливому функціоналі для користувачів.

Характеристику потенційних клієнтів стартап-проекту наведено у таблиці 6.5. Таблиця 6.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	**Потреба, що формує ринок**	**Цільова аудиторія (цільові сегменти ринку)**	**Відмінності у поведінці різних потенційних цільових груп клієнтів**	**Вимоги споживачів до товару**
1.	Необхідність побудови розумної системи IoT у себе в дома	Люди з технічною освітою, які прагнуть оптимізувати своє життя	Різні розміри об’єктів, різні мобільні пристрої,	Наявність веб інтерфейсу, віддаленого керування
2.	Використання «розумних» пристроїв у промисловості	Компанії та підприємства	Різні сервіси інтеграції, різні типи пристроїв	Наявність підтримки надійного протоколу зв’язку

З таблиці бачимо, що існує дві потенційні групи клієнтів. Для охоплення всіх цільових клієнтів, необхідна простота керування пристроями, а також надійний зв’язок між «розумними» речами. Таким чином проект задовольняє вимогам споживачів.

Надалі необхідно проаналізувати можливі загрози, та способи протидії. Фактори загроз наведено у таблиці 6.6.

Таблиця 6.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Конкуренція	Вихід на ринок великої компанії	1) Вихід з ринку 2) Запропонувати великій компанії поглинути себе 3) Передбачити додаткові переваги власного ПЗ для того, щоб повідомити про них саме після виходу міжнародної компанії на ринок
2.	Зміна потреб користувачів	Користувачам необхідне програмне забезпечення з іншим функціоналом	1) Передбачити можливість додавання нового функціоналу до створюваного ПЗ
3.	Економічні чинники	Рівень інфляції, зростання цін на апаратні пристрої	1) Розглянути можливість іншої апаратної реалізації
4.	Соціально-культурні чинники	Зміна напрямку розвитку галузі	1) Підкоритись умовам ринку
5.	Час розробки	Необхідно буде більше часу, ніж заплановано	1) Спростити деякі функції, щоби вчасно розробити проект

Основною загрозою для стартапу є конкуренція. Хоча існує небагато конкурентів на даний момент, ніша перспективна, а тому великі компанії можуть також звернути на неї увагу. Також з різким розвитком технологій необхідно враховувати мінливість користувачів, тобто бути готовими пристосуватись до зміни їх потреб.

Після запуску проекту, важливо продовжувати спостерігати за розвитком галузі, та реагувати відповідно, щоб і надалі нарощувати клієнтів. Фактори можливостей наведено у таблиці 6.6.

Таблиця 6.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Зростання можливостей потенційних покупців	Ріст зацікавленості до продукту серед інших груп користувачів з різним рівнем технічної грамотності	Додати підказки, інструкції та демонстрації роботи системи
2.	Зниження довіри до конкурента 3	У ПЗ конкурента №3 нещодавно була знайдена помилка, завдяки чому вдалося отримати контроль над системою третій особі	При виході на ринок звертати увагу покупців на безпеку нашого ПЗ
3.	Поява нових пристроїв	З'явились нові типи пристроїв	Реалізувати підтримку нових датчиків. Проведення маркетингової компанії, що вказує на підритку нових виробників або нових типів датчиків
4.	Поява нових апаратних платформ	Більш потужні мікрокомп'ютери та мікроконтролери	Перенести програмну реалізацію на новий тип мікрокомп'ютерів, що дасть можливість реалізувати додаткові функції (статистика, аналітика)
5.	Поява більш надійних безпровідних каналів зв'язку	На даний момент, безпровідні технології, вважаються менш надійними.	Додати програмну підтримку бездротових пристроїв

Основною дією для збільшення аудиторії, що користується нашим рішенням є простота користування інтерфейсом, а також надійність і безпека

даних. Відтак можлива потреба у дороблені програмного забезпечення для використання останніх технологій або для успішної взаємодії із новими пристроями.

Надалі виділяються особливості конкуренції на ринку. У таблиці 6.8 наведено ступеневий аналіз конкуренції на ринку.

Таблиця 6.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	**В чому проявляється дана характеристика**	**Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)**
1. Вказати тип конкуренції - досконала	Існує 3 фірми конкуренти на ринку	Врахувати ціни конкурентних компаній на початкових етапах створення бізнесу, реклама (вказати на конкретні переваги перед конкурентами)
2. За рівнем конкурентної боротьби - міжнародний	Компаній – з інших країни	Додати можливість вибору мови ПЗ, щоб легше було у майбутньому вийти на міжнародний ринок
3. За галузевою ознакою - внутрішньогалузева	Конкуренти мають ПЗ, яке використовується лише всередині даної галузі	Створити основу ПЗ таким чином, щоб можна було легко переробити дане ПЗ для використання у інших галузях
4. Конкуренція за видами товарів: - товарно-видова	Види товарів є однаковими, а саме – програмне забезпечення	Створити ПЗ, враховуючи недоліки конкурентів
5. За характером конкурентних переваг - нецінова	Вдосконалення технології створення ПЗ, щоб собівартість була нижчою	Використання менш дорогих технологій для розробки, ніж використовують конкуренти
6. За інтенсивністю - марочна	Бренди присутні	Активна реклама, яка вказує на переваги саме даного рішення, натякаючи на недоліки конкурентів

Після загального аналізу конкуренції, розглядаються У таблиці 6.9 наведено аналіз конкуренції в галузі за М. Портером.

Таблиця 6.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	**Прямі конкуренти в галузі**	**Потенційні конкуренти**	**Постачальники**	**Клієнти**	**Товаризамінники**
	Навести перелік прямих конкурентів	**Визначити бар'єри входження в ринок**	**Визначити фактори сили постачальників**	**Визначити фактори сили споживачів**	**Фактори загроз з боку замінників**
Висновки:	Існує 3 конкуренти на ринку. Найбільш схожим за виконанням є конкурент 3, так як його рішення також представлене у вигляді веб-додатку та мобільному додатку.	Є конкуренти, є можливість виходу на ринок	Постачальників мало	Важливим для користувача є наявність веб та мобільного додатку а також безпека системи	Товаризамінники можуть використати більш дешеву технологію створення ПЗ та зменшити собівартість товару.

Виходячи з аналізу можна сказати, що у проекту є можливості для входу на ринок. Ринок відкритий і постачальники не диктують правил, бо відсутні. Основні вимоги користувачів покриваються даною реалізацією, тому можна сказати, що проекту є шанси витримати конкуренцію.

Надалі необхідно оцінити наскільки фактори конкурентоспроможності а провести аналіз сильних та слабких сторін проекту. Основними сильними сторонами можна назвати надійність протоколу зв'язку та підтримку датчиків

різних виробників, адже реалізація конкурентів зроблена для роботи в інфраструктурі одного постачальника датчиків (табл. 11).

Таблиця 6.10 Обґрунтування факторів конкурентоспроможності

№ п/п	**Фактор конкурентоспроможності**	**Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)**
1.	Надійний протокол об’єднання пристроїв	Безпека та надійність «розумних» пристроїв відіграє важливу роль у виборі проекту користувачем
2.	Простота інтерфейсу користувача	Простота роботи х програмою спрощує роботу користувачеві, що робить її більш комфортною та зручною
3.	Підтримка різноманітних пристроїв	Можливість легкого підключення різних видів датчиків за допомогою єдиного протоколу
4.	Низька ціна рішення	Дозволяє охопити аудиторію, що не може собі дозволити рішення конкурента.

Можна сказати, що у стартап-проекту є достатньо факторів конкурентоспроможності, що надають йому переваги у боротьбі за споживачів. Також важливо відмітити, що простота інтерфейсу користувача та відкритість у роботі із різними пристроями є ключовими побажаннями майбутніх користувачів, адже це надає більше свободи у виборі.

У таблиці 6.11 наведено порівняльний аналіз сильних та слабких сторін стартап-проекту

Таблиця 6.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні						
			-3	-2	-1		1	2	3
1.	Надійний протокол для об’єднання пристроїв	20		+					
2.	Простота інтерфейсу користувача	15				+			
3.	Наявність додатків під різні платформи	10			+				

Із результатів можна зробити висновок, що рішення має як і значні переваги, такі як надійний протокол об’єднання та підтримка датчиків від різних виробників, так і свої недоліки, що проявляються у відсутності готової платформи. Проте фінальний продукт має бути конкурентоспроможним.

У таблиці 6.12 наведено SWOT- аналіз стартап-проекту.

Таблиця 6.12 – SWOT- аналіз стартап-проекту

Сильні сторони: Простота інтерфейсу користувача, захищеність, наявність веб та мобільного додатку	Слабкі сторони: відсутність мобільного додатку, розкрученості бренду
Можливості: у конкурента 3 виявлена проблема із безпекою ПЗ, зацікавленість продуктом ширших груп споживачів	Загрози: конкуренція, зміна потреб користувачів

У таблиці 6.13 наведено альтернативи ринкового впровадження стартаппроекту.

Таблиця 6.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	**Альтернатива (орієнтовний комплекс заходів) ринкової поведінки**	**Ймовірність отримання ресурсів**	**Строки реалізації**
1.	Використання RESTfull API для взаємодій пристроїв контролю параметрів	60%	1 місяці
2.	Використання MQTT для взаємодій пристроїв контролю параметрів	80%	2 місяць

З означених альтернатив обирається та, для якої: а) отримання ресурсів є більш простим та ймовірним; б) строки реалізації – більш стислими. Обираємо альтернативу 1, тому що вона має більшу ймовірність отримання ресурсів та менший час реалізації.

6.4 Розроблення ринкової стратегії проекту

У таблиці 6.14 наведено вибір цільових груп потенційних споживачів.

Таблиця 6.14 Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Люди з технічною освітою, що мають схильність та ентузіазм до покращення оточуючої середи	Критичним є інтеграція з іншими сервісами, можливість використання сторонніх пристроїв	Контроль параметрів приміщення, автоматизація процесів	Існує 3 конкуренти, які надають схожі рішення. До того ж – лише 1 конкурент надає веб та мобільний додаток	У сегмент увійти просто, необхідно лише надати можливість до самостійної зміни деяких компонентів
2.	Люди, які мають достатньо коштів і небайдужі до впровадження нових рішень	Критичним є простій та зрозумілий інтерфейс	Контроль параметрів приміщень		Маючи простий та зрозумілий інтерфейс, вийти на ринок не складно
3.	Люди, які хочуть економити енергоносії	Критичним є енергоефективність	Автоматизація процесів, енергозбереження		Просто, протокол спілкування між пристроями створений для енергозбереження

Виходячи із потреб різних цільових аудиторій, можна сказати, що характеристики фінальної реалізації найкраще підходять для власників будинків та квартир, бо мають низьку ціну, а також для підприємств, що висувають вимоги до витримки доволі екстремальних зовнішніх чинників (висока вологість та температура, постійні вібрації).

У таблиці 6.15 наведено визначення базової стратегії розвитку.

Таблиця 6.15 – Визначення базової стратегії розвитку

№ п/п	**Обрана альтернатива розвитку проекту**	**Стратегія охоплення ринку**	**Ключові конкурентоспроможні позиції відповідно до обраної альтернативи**	**Базова стратегія розвитку***
1.	Використання MQTT для взаємодій пристроїв контролю параметрів	Ринкове позиціювання	Простота використання, пришвидшення роботи, кросплатформеність	Диференціації

Для більш широкого охоплення користувачіва та IoT платформ, можливо додати протокол MQTT для покращення взаємодії між пристроями. Таким чином проекту буде надана відмінна особливість.

У таблиці 6.16 наведено визначення базової стратегії конкурентної поведінки.

Таблиця 6.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1.	Ні	Так	Буде, наявність веб сторінки	Зайняття конкурентної ніші

Проект не є «першопрохідцем», адже існують готові аналоги, але представлене рішення буде мати свої ключові особистості. Ринок, ще розвивається, тому нормальною стратегією буде пошук як нових клієнтів в конкурентній ніші, так само і забирати потенційних користувачів у інших компаній.

У таблиці 6.17 наведено визначення стратегії позиціонування.

Таблиця 6.17 Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартаппроекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1.	Простота інтерфейсу, можливість кастомізації системи	Диференціації	Простота користувацького інтерфейсу, що дозволяє спростити роботу з системою, можливість кастомізації, що вимагає наявність можливості роботи з сторонніми системами та пристроями	Кастомізація, простота, гнучкість

Вимоги цільової аудиторії співпадають із основними конкурентними якостями проекту. Стратегі розвитку полягає у спеціалізації на потребах власників квартир, а також у промисловості, та удосконалені реалізації функцій, які вони потребують, що і буде основною відмінністю від конкурентів.

6.5 Розроблення маркетингової програми стартап-проекту

У таблиці 6.18 наведено визначення ключових переваг концепції потенційного товару.

Таблиця 6.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	**Потреба**	**Вигода, яку пропонує товар**	**Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити**
1.	Кастомізація	Можливість інтеграції з іншими системами	Можливість роботи з модулями інших систем та пристроями
2.	Простота інтерфейсу	Простота та зручність ПЗ	Користувачам не потрібно замислюватися над тим як працювати з системою

У таблиці 6.19 наведено опис трьох рівнів моделі товару.

Таблиця 6.19 – Опис трьох рівнів моделі товару

Рівні товару	**Сутність та складові**
І. Товар за задумом	Об’єкт дозволяє користувачам налаштувати параметри моніторингу стану приміщення, керувати віддалено пристроями

II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Наявність веб додатку 2. Надійний канал зв'язку 3. Інтеграція з іншими сервісами 4. Простота використання	-	-
	Якість: згідно до стандарту ISO 29119 буде проведено тестування		
	Маркування присутнє.		
	Моя компанія. «IoTech»		
III. Товар із підкріпленням	Відсутня підтримка до продажу		
	Постійна підтримка для користувачів після продажу		
За рахунок чого потенційний товар буде захищено від копіювання: ліцензія.			

У таблиці 6.20 наведено визначення меж встановлення ціни.

Таблиця 6.20 – Визначення меж встановлення ціни

№ п/п	**Рівень цін на товаризамінники**	**Рівень цін на товарианалоги**	**Рівень доходів цільової групи споживачів**	**Верхня та нижня межі встановлення ціни на товар/послугу**
1	6000	7000	200000	5000

Таблиця 6.21 - Формування системи збуту

n/n	**Специфіка закупівельної поведінки цільових клієнтів**	**Функції збуту, які має виконувати постачальник товару**	**Глибина каналу збуту**	**Оптимальна система збуту**
.	Купляють готові апаратні рішення з програмним забезпеченням	Продаж	0(напряму), 1(через одного посередника)	Власна та через посередників

Таблиця 6.22 - Концепція маркетингових комунікацій

n/n	**Специфіка поведінки цільових клієнтів**	**Канали комунікацій, якими користуютьс я цільові клієнти**	**Ключові позиції, обрані для позиціонування**	**Завдання рекламного повідомлення**	**Концепція рекламного звернення**
.	Використання за допомогою сайту	Інтернет	Швидкодія, простота у використанні, безпека	Показати переваги сервісу, у тому числі і перед конкурента- ми	Демо-ролик із використання

6.6 Висновки

Провівши дослідження можна сказати, що проект можна комерціалізувати. Проект моє перспективи на ринк, бо бар'єрів майже не існує, ринок хаотично розвивається і не має монополістів та правил, що диктуються постачальниками. Також дана реалізація має значні переваги порівняно із конкурентами, хоча наявні і недоліки. Для успішного виконання проекту найкращим вибором є реалізація апаратної частини на Raspberry Pi із програмним забезпеченням написаним на мові програмування C++. Для успішного виходу на ринок та зайняття на ньому впевнених позицій, продукт повинен мати наступні характеристики:

- Мати низьку собівартість і як наслідок фінальну вартість
- Мати можливість працювати із датчиками різних виробників
- Надійність зв'язку між датчиками та шлюзом

Було проведено аналіз потенційних ризиків і можливостей, а також розраховані основні фінансово-економічні показники проекту. Отримані результати кажуть про те, що реалізація проекту є доцільною.

Було визначено сильні сторони проекту: наявність CAN, що дає змогу надійного спілкування між шлюзом і датчиками; використання оптимальних апаратних технологій для реалізації саме у будинках та квартирах. Серед слабких сторін проекту можна виділити відстутність хмарної платформи.

Можливості для створення конкурентного спроможного продукту включають зниження цін на апаратні запчастини та появу нових бібліотек, що спростять створення та підтримку програмного рішення. Серед загроз особливо важливими є різка зміна направленості ринку та поява нових лінійок конкурентів, що займають цільову бюджетну нішу. Менш небезпечними є політичні та економічні чинники, що можуть призвести до збільшення собівартості виготовлення апаратної частини продукту.

ВИСНОВКИ

Основним завдання даної дисертації було розроблення моделі узгодження протоколів керування в мережі CAN IoT та платформи ThingSpeak.

Для виконання поставленого завдання було проаналізовано архітектуру побудови IoT мережі. Розглянуто основні структурні шари, на які поділяють архітектуру IoT систем, серед них виділяють фізичний рівень – це актуатори, датчики, сенсори; мережевий рівень, на якому відбувається передача даних; рівень управління забезпечує безпеку та аналіз даних. Отже, проаналізовано їхні особливості та концепції, що дало змогу коректно побудувати модель.

Надалі, розглянуто протоколи, що використовуються для побудови мережі «розумних» речей. В цілому, вибір залежить від цілею проекту для якого будується система, але перевага надається протоколу CAN, через його надійність, достатню швидкість, низьке енергоспоживання. Серед протоколів передачі даних прикладного рівня, найбільш доцільно використовувати протоколи MQTT та REST.

Було розглянуто способи керування IoT пристроями, основними серед них є забезпечення та перевірка автентичності, конфігурація та контроль, моніторинг та діагностика, а також оновлення та технічне обслуговування програмного забезпечення. Оскільки CAN мережа працює тільки на фізичному і канальному рівні моделі OSI, для керування мережею «розумних» речей необхідне використання протоколів вищого рівня. Проаналізовано протоколи CANKingdom, CANOpen, SDS. Серед них CANOpen має найбільше переваг, так як існує вже багато доступних пристроїв, що підтримують цей стандарт, також специфікація гарно задокументована і наявні приклади реалізації. Таким чином було обрано CANOpen для побудови тестової моделі.

В рамках практичної частини було побудовано модель узгодження протоколів керування в мережі CAN IoT та платформи ThingSpeak. В якості IoT шлюзу було обрано Raspberry PI 3, який був об’єднаний в CAN мережу з

датчиком, побудованим на апаратній платформі Arduino Uno з сенсором температури та вологості. Реалізовано програму для узгодження протоколу CANOpen, який використовується між датчиком та шлюзом та HTTP Rest, що застовується для передачі даних на хмарний ThingSpeak через WiFi або Ethernet.

Результати роботи демонструють, що дане рішення в цілому відповідає поставленим завданням, а також що дана модель може використовуватись в мережах IoT, особливо, коли є вимоги до надійного зв’язку між пристроями, та низького енергоспоживання, адже в поняття «розумних речей» входять такі області застосування як промисловість, сільське господарство, медицина, торгівля, екологія, траспорт, безпека тощо.

ПРЕЛІК ПОСИЛАНЬ

1. Shimonski R. Network+ Study Guide & Practice Exams / Syngress, 2005.
2. Chung, J.-M. Internet Of Things & Augmented Reality Emerging Technologies.Yonsei University, 2017.
3. Fremantle, P. A Reference Architecture for The Internet of Things. London: WS02, 2015
4. Communication(Wireless) Protocols in IOT – Hardik Munjal – Medium – Режим доступу: https://medium.com/@hardy96tech/communication-wireless-protocols-in-iot-7da097ebbe96 – Дата доступу: 20.04.2018
5. Connectivity of the Internet of Things – Режим доступу: https://learn.sparkfun.com/tutorials/connectivity-of-the-internet-of-things – Дата доступу: 20.04.2018
6. Раскрываем тайны 6LoWPAN – Режим доступу: https://www.compel.ru/lib/ne/2015/11/5-raskryivaem-taynyi-6lowpan – Дата доступу: 20.04.2018
7. https://www.iotone.com/guide/the-iot-communication-protocols/g229
8. Fielding, R., & Reschke, J. 2014. The Internet Engineering Task Force (IETF Tools) – Режим доступу: https://tools.ietf.org/html/rfc7230#page-5. – Дата доступу: 20.04.2018
9. Wikibooks. 2015. – Режим доступу https://en.wikibooks.org/wiki/communication_networks/http_protocol. – Дата доступу: 20.04.2018
10. Fielding, Roy. Architectural Styles and the Design of Network-based / Software Architectures University of California, Irvine, 2000.
11. 5 Things to know about IoT platform – Режим доступу:https://iot-analytics.com/5-things-know-about-iot-platform/ – Дата доступу: 20.04.2018
12. Amazon — Режим доступу: https://aws.amazon.com/— Дата доступу: 20.04.2018

13. ThingSpeak How to analyze – Режим доступу: https://thingspeak.com/pages/how_to#analyze – Дата доступу: 20.04.2018
14. Fundamentals of IoT device management - IoT Design – Режим доступу: http://iotdesign.embedded-computing.com/articles/fundamentals-of-iot-device-management/ – Дата доступу: 20.04.2018
15. IDC: Worldwide Internet of Things Forecast, 2015–2020
16. Центр Интернета вещей Azure | Microsoft Azure – Режим доступу: https://azure.microsoft.com/ru-ru/services/iot-hub/ – Дата доступу: 20.04.2018
17. Pang Z., Chen Q., Tian J., Zheng L., Dubrova E. Ecosystem analysis in the design of open platform-based in-home healthcare terminals towards the internet-of-things // Proc. 2013, 15th Int. Conf. Adv. Commun. Technol. (ICACT). Korea, Pyeongchang.
18. Higher Layer Protocols – Режим доступу: https://www.kvaser.com/about-can/higher-layer-protocols/ – Дата доступу: 20.04.2018
19. Pang Z., Chen Q., Han W., Zheng L. Value-centric design of the internet-of-things solution for food supply chain: Value creation, sensor portfolio and information fusion // Inf. Syst. Front.
20. CanOpenSocket – Режим доступу: https://github.com/CANopenNode/CANopenSocket – Дата доступу: 20.04.2018
21. Ji Z., Qi A. The application of internet of things (IOT) in emergency management system in China // Proc. 2010 IEEE Int. Conf. Technol. Homeland Security (HST).